University of Alberta

ROBUST GRID-BASED DEPLOYMENT SCHEMES FOR UNDERWATER OPTICAL SENSOR NETWORKS

by

Abdullah Al Reza ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-47399-3
Our file Notre référence
ISBN: 978-0-494-47399-3

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Underwater sensor networks have received significant attention from the research community in recent years. Since radio signals face excessive absorption in the underwater environment, acoustic communication has been the dominant physical layer medium in the literature. Although acoustic communication has long range and omni-directional characteristics like terrestrial radio, it suffers from excessive propagation delay in water and very low bandwidth. In this thesis, we consider the design of an optical underwater sensor network based on low cost LEDs and photodiodes. Such an optical communication system has shorter range compared to acoustic systems but is cheaper and lighter and, most importantly, can support significantly higher data rates. Optical communication is characterized by the line of sight property which makes optical links vulnerable to occasional failures due to underwater organisms and moving particles. We consider a grid based deployment of underwater sensor nodes and the selection of a topology based on a minimal set of point-to-point optical links that is robust to occasional link failures. We develop patterns for networks with maximum 1, 2 and 3 interfaces per node constraints. We evaluate the robustness of our proposed deployment patterns by simulating three simple resilient routing protocols on these patterns and demonstrate that our patterns support a high degree of robustness even though they use only a fraction of all potential links in the grid graph, thereby minimizing the cost of deployment.

Acknowledgements

I would like to thank my supervisor Dr. Janelle Harms for her support and guidance throughout the course of my thesis. She has always managed to find time to review my work, listen to my problems and throw valuable comments and suggestions. She has also helped me find the topic for this thesis. Above all, she has been a mentor for me throughout my graduate student life at the University of Alberta and guided me in numerous ways to become a successful graduate student. I would also like to thank Dr. Lorna Stewart for reviewing my work and providing valuable insights and suggestions. I would like to express my thanks to numerous friends at the department who have been there for me throughout the course of my thesis. I would like to particularly mention Sunil Ravinder, Israat Haque, Abhishek Srivastava, Ananth Venkateswaran, Chen Liu and Jianzhao Huang for their support. I would also like to thank my friend Susanjib Sarkar from Mechanical Engineering for being a true friend to me throughout my thesis. Finally, I would like to thank my parents for their love and their support.

Table of Contents

1 Introduction **1**
  1.1 Motivations 1
  1.2 Contributions: Robust Grid-based Deployment Schemes for Optical UWSN 3
  1.3 Layout of the Thesis 5

2 Related Work **7**
  2.1 Underwater Acoustic Sensor Networks 7
  2.2 Underwater Optical Sensor Networks 12
    2.2.1 Optical Model used in this Thesis 15
  2.3 Sensor Network Deployment 17
  2.4 Chapter Summary 19

3 Robust Grid-based Deployment Topologies **20**
  3.1 Problem Definition 20
    3.1.1 Optical Interface Model 23
  3.2 Robust Deployment Schemes 24
    3.2.1 Maximum 1 Interface per Node 24
    3.2.2 Maximum 2 Interfaces per Node 26
    3.2.3 Maximum 3 Interfaces per Node 27
  3.3 Static Evaluation of Proposed Deployment Topologies 37
    3.3.1 Failure Models 39
    3.3.2 Results 40
  3.4 Chapter Summary 45

4 Dynamic Evaluation of Deployment Topologies **46**
  4.1 Routing Protocols 47
    4.1.1 Flooding (FLD) 47
    4.1.2 Dual Paths Protocol (DPP) 48
    4.1.3 Hop-by-Hop Acknowledgment with Local Update (HHA) 49
  4.2 Simulation Environment 54
    4.2.1 Traffic Model 55
    4.2.2 Error Blobs 55
  4.3 Simulation Metrics 56
    4.3.1 Delivery Ratio 56
    4.3.2 Average Delay Per Packet (ADPP) 57
    4.3.3 Average Number of Payloads Transmitted per Successful Packet (APTS) 57
  4.4 Experiment Methodology 58
    4.4.1 Assumptions 58
    4.4.2 Parameter Settings 59
    4.4.3 Design of Experiments 60
  4.5 Analysis of Experimental Results 61
    4.5.1 Analysis of Flooding Protocol (FLD) 61
    4.5.2 Analysis of Dual Paths Protocol (DPP) 73
    4.5.3 Analysis of Hop-by-Hop Acknowledgment Protocol (HHA) 80
  4.6 Possible Effects with Multiple Sources 92
  4.7 Chapter Summary 94

5 Conclusion and Future Work **95**
- 5.1 Conclusions 95
- 5.2 Future Directions for Research 97

Bibliography **99**

A Implementation of FLD, DPP and HHA **103**
- A.1 Implementation of Flooding (FLD) 103
- A.2 Implementation of Dual Paths Protocol (DPP) 104
- A.3 Implementation of Hop-by-Hop Acknowledgment Protocol (HHA) 105
- A.4 Packet Formats 106

B Verification and Validation of Simulation Model **108**

C Pseudo-code of Event Handlers for the Simulation Models **113**
- C.1 Event Handlers for FLD Protocol 114
- C.2 Event Handlers for HHA Protocol 116

List of Tables

3.1 Summary of proposed deployment topologies 38
3.2 Properties of different topologies when applied on a 12x12 grid with the sink at center 38

4.1 Simulation Configurations 55
4.2 Performance of FLD, DPP and HHA on TOP6 at a hop distance of 10 hops (a = 20m, b = 4m, speed = 15 cm/sec and TTL = 0.5 sec for HHA) 89
4.3 Performance of FLD, DPP and HHA on TOP6 with error blob dimension a = 30m, b = 4m (hop distance=10 hops, speed=15 cm/sec and TTL=0.5 sec for HHA) 92

A.1 Different fields and their sizes in a fixed-length DATA packet of FLD/DPP protocol 107

List of Figures

3.1 Grid-based deployment problem 21
3.2 Optical interface model: (a) maximum one interface per node (b) maximum 2 interfaces per node (c) maximum 3 interfaces per node 24
3.3 Topologies with 1-interface constraint: (a) One directed Hamiltonian Cycle (b) 4 directed Hamiltonian cycles (TOP1) 26
3.4 TOP2: 4 undirected Hamiltonian cycles 27
3.5 Manhattan Distance Property 29
3.6 Four quadrants around the sink: edges on the axes cannot be avoided in a shortest path spanning tree from the sink 30
3.7 A quadrant with horizontal axis of length $m = 7$ and vertical axis of length $n = 8$ 31
3.8 Optimal pattern for a quadrant: (a) $x < y$: total 3-degree nodes $= (x - 2)$ (b) $y < x$: total 3-degree nodes $= (y - 2)$ 32
3.9 (a) Optimal patterns for individual quadrants lead to optimal pattern for the entire grid (b) Optimal patterns for individual quadrants violate the 3-degree constraint 33
3.10 Pattern using only the vertical axis (clockwise right axis) to place 3-degree nodes 34
3.11 (a) TOP3: pattern from the corollary applied on 4 quadrants (b) TOP4: Leaves are connected 36
3.12 (a) TOP5: Q_1&Q_4 and Q_2&Q_3 are connected (b) TOP6: Q_1&Q_2 and Q_3&Q_4 are connected 36
3.13 An "error blob": an ellipse representing an underwater obstacle 40
3.14 Robustness to isolated failure 41
3.15 Average path length: isolated failure 42
3.16 Robustness to patterned failure with an elliptical failure model. $b = 4$m and $\lambda = 3$ 44
3.17 Average path length under elliptical failure model. $b = 4$m and $\lambda = 3$ 44

4.1 Routing around the error blob in HHA 54
4.2 Delivery ratio vs hop distance with FLD. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 64
4.3 ADPP vs hop distance with FLD. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 64
4.4 APTS vs hop distance with FLD. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 65
4.5 Delivery ratio vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec 69
4.6 ADPP vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec 70
4.7 APTS vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec 70
4.8 Delivery ratio vs error blob speed with FLD. $a = 20$m, $b = 4$m and Hop distance = 10 72
4.9 ADPP vs error blob speed with FLD. $a = 20$m, $b = 4$m and Hop distance = 10 72
4.10 APTS vs error blob speed with FLD. $a = 20$m, $b = 4$m and Hop distance = 10 73
4.11 Delivery ratio vs hop distance with DPP. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 76
4.12 ADPP vs hop distance with DPP. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 76

4.13 APTS vs hop distance with DPP. $a = 20$m, $b = 4$m and blob speed = 15cm/sec 77
4.14 Delivery ratio vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec 79
4.15 ADPP vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec 79
4.16 APTS vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec 80
4.17 Delivery ratio vs TTL with HHA. $a = 20$m, $b = 4$m, hop distance = 10 and blob speed = 50cm/sec 83
4.18 ADPP vs TTL with HHA. $a = 20$m, $b = 4$m, hop distance = 10 and blob speed = 50cm/sec 83
4.19 APTS vs TTL with HHA. $a = 20$m, $b = 4$m, hop distance = 10 and blob speed = 50cm/sec 84
4.20 Delivery ratio vs hop distance with HHA. $a = 20$m, $b = 4$m, blob speed = 15cm/sec and TTL = 0.5 sec 87
4.21 ADPP vs hop distance with HHA. $a = 20$m, $b = 4$m, blob speed = 15cm/sec and TTL = 0.5 sec 88
4.22 APTS vs hop distance with HHA. $a = 20$m, $b = 4$m, blob speed = 15cm/sec and TTL = 0.5 sec 88
4.23 Delivery ratio vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec 91
4.24 ADPP vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec 91
4.25 APTS vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec 92

B.1 Modules in the simulation implementation 109

Glossary of Terms

UWSN. Underwater Wireless Sensor Network.

MAC. Medium Access Control. The MAC layer in the communication protocl stack is responsible for dealing with the channel contention problem.

Optimal Pattern. A 3-degree constrained shortest-path spanning tree in a grid graph rooted at a given grid node with minimum number of 3-degree nodes.

LB. Lower Bound on the number of 3-degree nodes in an optimal pattern for a grid rooted at a given grid node. It is not possible to form a 3-degree constrained shortest-path spanning tree in a grid graph with less than LB number of 3-degree nodes.

Error Blob. An underwater obstruction modeled as an ellipse that blocks optical communication between two nodes by obscuring the line of sight.

TOP1. Four directed Hamiltonian cycles spanning all nodes in the grid. Each quadrant around the sink is spanned by one directed Hamiltonian cycle.

TOP2. Four undirected Hamiltonian cycles spanning all nodes in the grid. Each quadrant around the sink is spanned by one undirected Hamiltonian cycle.

TOP3. A 3-degree constrained shortest-path spanning tree in the grid rooted at the sink with (LB+2) 3-degree nodes in the worst case.

TOP4. TOP3 with leaves in each quadrant connected together by a path.

TOP5. TOP4 with quadrant 1&4 and quadrant 2&3 connected with two additional boundary edges.

TOP6. TOP5 with quadrant 1&2 and quadrant 3&4 connected with two additional boundary edges.

FLD. The Flooding protocol for routing.

DPP. The Dual Paths Protocol for routing.

HHA. The Hop-by-Hop Acknowledgment with local update protocol for routing.

ADPP. Average Delay Per Packet. This simulation metric indicates the delay a packet faces between its generation and delivery on average.

APTS. Average number of Payloads Transmitted per Successful packet. This simulation metric indicates the communication overhead incurred by a routing protocol applied on a particular deployment topology.

Chapter 1

Introduction

A wireless sensor network [10] is a collection of sensor nodes which are deployed in a target region with a primary task of sensing data and communicating with each other to make the data available to the user. In the most common settings, a special node in the network is designated as the *sink* node to which all other sensor nodes send their sensed data, possibly through other intermediate sensor nodes. Underwater wireless sensor networks (UWSN) [1] are a relatively new family of wireless sensor networks which are deployed under water, usually in a sea or an ocean, in order to monitor underwater environment. UWSNs make it possible to monitor underwater organisms, environment and other things without having to physically go under the sea and thus bring about enormous possibilities for research in various fields that make use of the oceanographic data. Above all, UWSNs allow us to monitor the underwater environment which constitutes 70% of the earth's surface.

1.1 Motivations

As a new and evolving field, UWSN has received a great amount of attention from the research community in recent years. Since radio and most other electromagnetic signals are excessively absorbed by water, acoustic communication has been considered almost exclusively for UWSN. While acoustic signals have a range of several hundred meters [1] and have omni-directional communication characteristics like terrestrial radio, they suffer from low bandwidth of 5 to 10 Kbps [49, 13], very long propagation delay of nearly 1500m/s [37], high and unpredictable error rate of acoustic signals in underwater environment [44, 40, 58] and high cost of acoustic modems [49, 38].

The long propagation delay and extremely low bandwidth, added with high and unpredictable error rates, make acoustic communication unsuitable for applications that have high bandwidth and short delay requirements, *e.g.*, real-time and multimedia sensing applications [46]. In this thesis, we consider an underwater sensor network where nodes communicate optically using light and low-cost light emitting diodes (LED) and photodiodes operating in green/blue visible range of spectrum. Such an optical system has extremely fast propagation and can support a bandwidth of several megabits per second [49, 13] compared to only a few kilobits per second supported by an underwater acoustic system. This high bandwidth and fast propagation make optical communication suitable for applications with real-time and high data rate requirements.

The price to pay for this achievement in bandwidth and propagation speed is the reduction in communication range. Depending on the clarity of water, the transmission power, presence of concentrator lenses at the transmitter and amplifiers at the receiver, the maximum communication range using such system varies from 8m to 40m [49, 13]. In contrast, acoustic communication can support ranges of several hundred meters, though energy-efficiency and reasonable speed calls for a range of less than 150m [18, 43]. The range of optical communication depends largely on the clarity of water. It has been demonstrated that ocean areas that are not in close proximity with coastal areas usually have clarity very close to pure water [43]. Thus, for an underwater sensing application that has a stringent requirement for high bandwidth and has a smaller area of observation that is not very close to the coastal areas, a point-to-point optical communication system using low cost LEDs and photodiodes is an attractive solution.

Because of the limited range and directional communication characteristics, connectivity becomes an important design goal in such an underwater optical sensor network. Unlike acoustic UWSN where nodes have long and omni-directional range of communication, connectivity in an optical UWSN is not inherent in the deployment and calls for careful placement of nodes in the target sensing region so that the resulting deployed network possesses the desired level of connectivity. In addition, since optical transceivers are directional, higher degree of connectivity calls for higher number of transceivers in the network which introduces a new design goal of balancing the cost of deployment (number of optical transceivers) with the degree of connectivity.

In this thesis, we design and evaluate deployment schemes for underwater optical sensor networks that balance the cost of deployment and the degree of connectivity desired in the network. Since optical communication depends on the line of sight property, underwater optical links are expected to go down occasionally due to obstructions like underwater organisms, floating objects and sediments. Therefore, we design deployment schemes that possess enough redundancies in order to support desired levels of robustness to occasional failures of optical links in the network.

1.2 Contributions: Robust Grid-based Deployment Schemes for Optical UWSN

In this thesis, we design two-dimensional grid-based deployment schemes for underwater optical sensor networks that support desired levels of robustness and path quality with minimum number of optical interfaces. We assume that the nodes and the sink are placed at grid points and then select point-to-point optical links between adjacent grid points to produce a connected and robust topology. Each link in the deployment topology introduces one optical interface for both nodes at the two ends of the link. Therefore, while designing our deployment topologies, we introduce the least number of optical links in the network to minimize the cost of deployment.

Since each optical interface in a node incurs an extra cost and requires extra space in the node, we place constraints on the number of interfaces a node can have. We consider three cases where each node in the network is constrained to have no more than 1, 2 and 3 optical interfaces. For each of these cases, we design deployment patterns that result in topologies with desired degrees of robustness and path quality with a close-to-minimum number of per node and total optical interfaces in the network. In order to ensure deterministic robustness, we design 2-edge-connected [53] topologies whenever the interface constraint allows us to do so in order to ensure that any arbitrary link in the network can go down without disconnecting any node from the sink. For 3 interfaces per node constraint, we design deployment topologies in which a shortest path from sink to each sensor node in the network is available in order to support lowest-cost communication to and from the sink in a failure-free environment. For this constraint, we minimize the number of 3-degree nodes in the network since these nodes are expensive and contribute in increasing the total number of interfaces in the network.

We perform two kinds of simulation-based evaluation in order to analyze the performance of our deployment topologies. First, we perform a static evaluation where we do not consider dynamic behavior of the network such as as variation of failure rates with time, movement of link-blocking obstacles with time, dynamic selection of routing paths to avoid obstacles and network operations like routing, fault-detection etc. that changes with time. We apply isolated and patterned failure models on our topologies and evaluate probabilistic robustness of these topologies to these failures and the quality of available paths when some links in the network are down. Then we perform a dynamic evaluation where we apply three simple resilient routing protocols on our deployment topologies and evaluate their performance in terms of resiliency to link failures, average delay of packet delivery to the sink and overall communication overhead in the presence of multiple moving obstacles inside the network.

The contributions of our thesis can be summarized as follows:

- Deployment topologies for optical UWSN that utilize four directed and undirected Hamiltonian cycles in a grid for cases with 1 and 2 interfaces per node constraints, respectively.

- The formulation pattern for a 3-degree constrained shortest path tree in a grid rooted at the sink and spanning all nodes in the grid with (LB+2) number of 3-degree nodes in the worst case where LB is the lower bound on the number of 3-degree nodes in such a tree. Our proposed formulation pattern works for any grid dimension and any placement of the sink inside the grid.

- A series of deployment patterns built on the 3-degree constrained shortest path spanning tree that support increasingly higher degrees of robustness by adding additional links in the network at strategic points. These topologies have shortest paths from the sink to all grid nodes since they are built on a shortest path tree while at the same time they support higher degrees of robustness by strategically introducing higher degrees of redundancies in the network.

- A static simulation-based evaluation of the proposed deployment topologies that evaluates and compares the probabilistic robustness and path qualities of these topologies by applying isolated and patterned failure models on the them.
- A dynamic evaluation of the proposed topologies performed by simulating three simple resilient routing protocols on these topologies with one packet generating source and multiple moving obstacles in the network. The performance of these routing protocols on our topologies has been evaluated and compared in terms of packet delivery ratio, average delay of delivery of packets to the sink and overall communication overhead.
- Directions for future research based on the research in this thesis.

1.3 Layout of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we present the infrastructure and research challenges in acoustic underwater sensor networks and how underwater optical communication differs from underwater acoustic communication. We also present underwater optical communication characteristics and models and discuss the problem of sensor network deployment in terrestrial radio and acoustic underwater environment and how it differs from the deployment problem with underwater optical sensor network.

In Chapter 3, we present the formal definition of the problem that we consider in this thesis and the design goals that we try to achieve. Then we design robust grid-based deployment topologies for three cases where no node in the network is allowed to have more than 1, 2 and 3 interfaces. Finally, we evaluate the probabilistic robustness of our deployment topologies by simulating them with isolated and patterned failure models. We do not consider dynamic aspects of network operations in this static evaluation.

In Chapter 4, we perform detailed dynamic evaluation of our deployment topologies by simulating three simple resilient routing protocols on these topologies. We use a single packet generating source and multiple moving obstacles in the network and evaluate the performance of these protocols applied on our topologies in terms of packet delivery ratio (resiliency), average delay of delivery and communication overhead.

In Chapter 5, we summarize the findings and contributions of our thesis. We also present directions for future research in this chapter.

Chapter 2

Related Work

In this chapter, we present the research work in the current literature that are related to our thesis. First, we present the architecture and research challenges for acoustic underwater sensor networks since acoustic signaling remains the dominant means of communication for underwater sensor networks in the current literature. Then we present research work on underwater optical signaling and underwater sensor networks that make use of optical communication to achieve higher bandwidth. Finally, we present related work on the deployment of sensor nodes in a target field to achieve particular design goals like coverage and connectivity.

2.1 Underwater Acoustic Sensor Networks

Akyildiz *et al.* present two generic architectures for underwater sensor networks [1]. The first architecture is a two-dimensional architecture where nodes are placed on the sea-floor with the help of anchors. Possible applications of such networks include monitoring the environment and organisms at the ocean bottom, monitoring underwater plates in tectonics etc. The second architecture is a three-dimensional architecture where nodes are placed at different depths with the help of anchors that pull them downwards through wires and attached buoys that pull them upwards. The length of the wire, preferably electronically controlled by the node, determines the depth of the node. Possible applications include monitoring organisms and environment at different depths, monitoring pollution, ocean currents etc.

Akyildiz *et al.* [1] also describe the unique characteristics of underwater nodes that separate them from traditional terrestrial sensor nodes and make the design goals different in an underwater environment regardless of the physical layer used for communication. Underwater sensor nodes are more expensive than terrestrial nodes because of the need for water-proof casings, special techniques for keeping them afloat, if applicable, etc. This high cost of nodes results in a relatively low density of nodes in underwater sensor networks compared to that in the terrestrial counterparts. The high cost of nodes often implies that placing nodes randomly in the target area in large numbers is not a feasible solution. Rather, careful deployment of nodes become more important in underwater sensor networks which is the main focus of our thesis. Other differences between underwater and terrestrial sensor nodes include lack of spatial correlation among sensed data from neighboring nodes because of the increased distance between them and higher cost of communication resulted from increased distance among nodes and sophisticated signaling techniques which is especially true for acoustic communication.

Since radio and most other electromagnetic signals are excessively absorbed by water, acoustic communication has been considered almost exclusively for underwater sensor networks (UWSN). Acoustic communication has desirable similarities with terrestrial radio in that it has omni-directional communication characteristics like radio. In addition, underwater acoustic communication supports a range of several hundred meters [1]. The downsides are the very low bandwidth of 5-10 Kbps [49, 13], very long propagation delay of nearly 1500 m/s [37], high and unpredictable error rates of acoustic signals in underwater environments [44, 40, 58] and high cost of acoustic modems [49, 38]. Communication protocols at different layers in an acoustic UWSN try to deal with the problems of low bandwidth, slow propagation and high error rates.

Low bandwidth and extremely slow propagation introduce considerably higher number of collisions in medium access control (MAC) layers for acoustic UWSN and degrades the overall performance significantly if not handled well. Therefore, MAC layer protocols for acoustic UWSN [34, 57, 32] primarily focus on reducing the number of collisions in order to prevent the wastage of the inherently poor channel capacity in such networks. For example, Paleato *et al.* [34] propose the use of a new warning packet along with the RTS/CTS scheme in the original MACA protocol [20] designed for terrestrial radio. Because of the very long propagation delay, a node that just received an RTS and replied with a CTS (Clear To Send) may receive another RTS from a third node which was transmitted quite a while ago. In such cases, the node that received the second RTS may send a warning packet to the originator of the first RTS (Request To Send) to keep it from transmitting and causing collision. A node receiving a CTS does not start transmission right away. Rather, it waits for a while in case a warning packet arrives in which case it aborts transmission. The R-MAC protocol from Xie *et al.* [57], on the other hand, uses careful scheduling to avoid collisions altogether. The scheduling scheme is designed to improve energy efficiency and ensure fairness of channel access among different nodes.

Routing protocols for acoustic UWSN can primarily be classified into two groups, each group having the common goal of minimizing the consumption of energy which is a scarce resource in sensor networks. The first group of routing protocols [37, 18] focus on the use of the acoustic channel intelligently to fully utilize the limited channel capacities resulting from low bandwidth, long propagation and high error rate. Pompili *et al.* [37] use an acknowledgment-based forwarding method to achieve reliability and observe that channel utilization efficiency in underwater acoustic environment is unacceptably low under such circumstances which also results in high energy consumption. To improve efficiency, they propose the use of packet trains where a group of packets are forwarded in a row and one acknowledgment is sent for each train instead of one acknowledgment for each packet. The acknowledgment indicates which packets, if any, were corrupted in the train so that the sender can include those packets in the next train. Thus, channel efficiency is decoupled into train efficiency and packet efficiency. Since trains are never retransmitted, we can use very long trains to improve overall channel efficiency. Packets, on the other hand, are retransmitted and therefore, we need to select an optimal size of each packet to achieve the maximum efficiency. The authors also demonstrate the use of forward error correction codes (FEC) to improve the efficiency further and propose two geographic routing schemes that make use of the above observations.

Harris III *et al.* [18] thoroughly examine the physical layer characteristics of underwater acoustic media in order to deduce practical design goals for routing algorithms in acoustic UWSN. They use underwater acoustic communication models to find out that unlike terrestrial radio where attenuation is mainly a function of distance, the attenuation of underwater acoustic signals is a function of both distance and frequency and an increase in any or both of these factors results in an increase in attenuation for underwater acoustic signals. The result is that as we decrease the distance between communicating nodes, not only is the attenuation reduced but also the bandwidth available for communication improves. The authors use a commercially available acoustic modem to deduce the relationship of hop distance with available bandwidth and energy consumption and find out that a hop distance of approximately 150m provides the optimum result for the modem in hand. Although the actual value of this distance is different for different modems, this work provides a guideline to finding out the desired hop distance for a particular communication device and suggests that routing algorithms for acoustic UWSN should try to select hops with lengths close to this distance instead of blindly minimizing the number of hops or per hop distance in the selected routes. Harris III *et al.* [18] also propose a geographic routing protocol called Bounded Distance that makes use of the above fact and tries to select hops that have lengths close to 150m. Their simulation shows that Bounded Distance reduces overall energy consumption compared to protocols that try to minimize the number of hops or per hop length of the selected routes.

The second and the most thoroughly investigated group of routing protocols for acoustic UWSN consists of the protocols that focus on improving the reliability and resiliency of delivery in underwater acoustic environment that is characterized by high and unpredictable error rates [44, 40, 58, 36, 39, 29]. Sun *et al.* [44] summarize different sources of noise that result in high and unpredictable error rate in underwater acoustic channels. These noises include man-made noise, ambient noise and noises caused by medium's physical characteristics. Man-made noises include acoustic disturbance caused by ships and other water vehicles. Ambient noises include wave motion, storms on surface, underwater organisms etc.

Most of these resilient routing schemes for acoustic UWSN handles the problems of resiliency and reliability by using multiple paths for delivering a packet. Sun *et al.* [44] propose a scheme called Packet Cloning where a belt-like route is formed from source to sink and multiple copies of a packet (clones) are transmitted by the source with short intervals.

Packets are forwarded along the belt and the broadcast property of acoustic transmission ensures that all or most possible paths through the belt forward a clone even though some links are broken. Also, if a clone is lost, forwarding nodes can reproduce the clone by copying subsequent clones of the same packet which ensures even higher resiliency. The Vector-Based Forwarding (VBF) from Xie *et al.* [58] also uses a belt-like route to achieve high resiliency but it works in a different geographic manner. Given that the source and the sink know their locations, a routing belt consists of all the nodes that are within a predefined distance from the vector connecting the source and the sink. Forwarding nodes calculate their locations on the fly from the locations of the preceding nodes and the reception angle. This eliminates the need for a separate localization scheme and supports node mobility.

While the above two protocols utilize the geographic proximity of forwarding nodes to achieve high degree of resiliency, failure may occur if all links in a particular area go down simultaneously. To this end, Seah *et al.* [40, 39] propose Virtual Sink architecture where multiple sinks are placed at geographically distant points in the network and this set of physical sinks is considered as one virtual sink in the sense that delivering a packet to any one of these physical sinks means a successful delivery to the virtual sink. They propose a multi-path routing protocol that forwards a packet simultaneously on multiple paths, one path for each sink, and present analytical results on the number of sinks that need to be deployed to achieve higher degrees of resiliency without incurring unreasonable cost. The resilient routing scheme proposed by Pompili *et al.* [36] uses Integer Linear Programming to find out two paths from each source to the (single) sink that incur minimum transmission energies and this information is passed to all nodes during setup phase. Packets from a source are forwarded on only one of these two paths but the selection of the actual path is made based on the dynamic behavior of the paths measured from ACK timeouts.

Lee *et al.* [29] propose a routing scheme called Underwater Diffusion (UWD) that is designed particularly for mobile sensor nodes that can move together in groups due to ocean current. Their primary goal is to avoid the need for flooding to support mobile routing. The idea behind UWD is to form a community of nodes around a forwarding node that is in the path from a source to the sink. Each community node of a forwarding node can hear transmissions from the forwarding node and the predecessor and successor of the forwarding node in the path from the source to the sink. As a result, if a forwarding node dies or moves away, its community nodes can detect it and one of the community members can become the new forwarding node, thus supporting mobility and robustness without flooding. Refresh packets are sent from source to the sink periodically along the path to reconstruct the communities on the face of mobility. Although UWD primarily handles mobility, robustness and resiliency is inherent in its design.

2.2 Underwater Optical Sensor Networks

As demonstrated in Section 2.1, although underwater acoustic signals support long ranges and have desirable omni-directional characteristics like terrestrial radio, they suffer from extremely low bandwidth, very long propagation delay and high and unpredictable loss rates. As a result, acoustic communication cannot support applications that have high bandwidth and short delay requirements, *e.g.*, real-time and multimedia applications. To overcome these shortcomings of underwater acoustic communication, underwater optical communication systems built with light and low cost LEDs and photodiodes operating in green/blue visible spectrum have been considered [49, 43, 13, 38] since only visible light in this spectrum propagates well in water. Such a communication system has a very high bandwidth and extremely fast propagation compared to an underwater acoustic system. Vasilescu *et al.* [49] experiment with their underwater optical system with a bit rate of 320 Kbps whereas Giles *et al.* [13] use a mathematical model of a commercially available LED and photodiode to produce a bit rate of up to 4.4 Mbps. This high bandwidth and fast propagation of light makes such a communication system suitable for applications that have real-time and high data rate requirements.

The downside of using such a communication system is the reduction in the range of communication compared to acoustic counterparts. Depending on the clarity of water, the transmission power, presence of concentrator lenses at the transmitter and amplifiers at the receiver, the maximum communication range using such system varies from 8m to 40m [49, 13]. In contrast, acoustic communication can support ranges of several hundred meters, though energy-efficiency and reasonable speed calls for a range of less than 150m [18, 43]. The range of optical communication decreases as the water becomes turbid. However, ocean areas that are not very close to the coastal areas usually have clarity very close to pure water [43] allowing larger communication ranges to be supported. Farr *et al.* [11] experiment with photomultiplier tubes as receivers instead of photodiodes to create a range as long as 100m in clear water. However, photomultipliers are costly, heavy and power consuming and therefore, not suitable for use with underwater sensor nodes. For this reason, we do not consider such devices in our thesis. However, these devices can be used in the sink which is naturally assumed to be more expensive or in networks where a small number of expensive nodes are deployed for long time observation. The design we present in the next chapter does not assume any specific value for the range and is therefore applicable to any configuration.

An example of an underwater sensor network that makes use of optical communication to achieve higher bandwidth is the hybrid network designed and implemented by Vasilescu *et al.* [49] where nodes have both acoustic and optical communication systems. The optical communication is performed using commercially available light emitting diodes (LEDs) as transmitters and photodiodes as receivers, both working in the green/blue region of the visible light spectrum, with additional concentrator lenses and amplifiers to improve the range. Acoustic communication is used by the nodes to talk to each other to perform localization. An intelligent autonomous underwater vehicle (AUV) periodically visits each node and downloads its sensed data using short-range and high-speed optical communication. Although this strategy supports high-speed data retrieval, it introduces long delay of data delivery because of the slow physical movement of the AUV which makes such system unsuitable for real-time applications. Also, this strategy causes high energy consumption resulting from the movement of the AUV.

The oceanographic contamination detection system proposed by Kedar *et al.* [22], on the other hand, uses only directional optical media to communicate under water and a static sink instead of a mobile AUV. The authors propose a distributed cluster of underwater sensor nodes that detect contamination levels of ocean water by means of optical backscattering [22] from contaminant particles. Because of the short range of underwater optical communication, they propose the deployment of a number of sensor nodes within the communication range of the sink where all sensor nodes transmit directly to the sink.

To minimize the interference at the sink caused by simultaneous transmissions from a large number of sensor nodes, the authors propose the use of Space Division Multiple Access (SDMA) [22] technique. In order to extend the area of observation, the authors propose the use of multiple such clusters at different locations with the sinks of different clusters communicating with each other to provide inter-cluster connectivity. However, the authors do not discuss the means of this long range communication among sinks.

The use of directional radio antennas instead of traditional omni-directional antennas has been investigated in the literature of terrestrial ad hoc and sensor networks [5, 27, 8, 4, 16, 6, 7, 17]. These antennas behave in a similar way to the optical transceivers considered in this thesis in that the transmitter forms a directional beam and the receiver needs to be in the sector covered by the beam. One difference is that most of this research work on directional radio antenna assume the presence of an omni-directional receiver that can receive signals from any direction. In contrast, optical receivers are directional and can receive only from a direction from which it is expecting a signal. Also, an assumption behind all this research work on directional radio antenna is that nodes are already deployed randomly in large and redundant numbers and the focus is thus on network operation and communication protocols. With underwater nodes, deploying nodes randomly in large numbers becomes expensive because of the high cost of underwater sensor nodes. Also, because of the limited range of optical communication, maintaining connectivity among nodes becomes an important design goal.

Wireless optical communication has been used in terrestrial sensor, ad hoc and mesh networks [15, 31, 21, 33]. These networks are referred to as Free Space Optical (FSO) networks and enjoy very long range of laser or infrared communication (up to several kilometers [33]) which is impossible in an underwater environment since laser and infrared signals are excessively absorbed by water [38]. Because of this long range, a node in a terrestrial FSO network can potentially connect with any other node but should choose to communicate only with an intelligently selected subset of nodes so that the resulting energy consumption and probability of line of sight obscuration is minimized. Therefore, design goals with terrestrial FSO networks are significantly different from the design goals with underwater optical sensor networks. Because of the limited communication range, node placement becomes the most important factor in underwater optical networks in order to maintain connectivity.

2.2.1 Optical Model used in this Thesis

In this thesis, we consider underwater sensor nodes with optical transceivers built with light and low-cost LEDs and photodiodes operating in green/blue visible spectrum. We use the underwater optical communication model proposed by Giles *et al.* [13]. In this model, the transmitter sends a light beam the width of which can be controlled by a concentrator lens placed with the LED. A narrow beam means longer range but calls for finer tuning with the receiver's line of sight. The receiver can receive the signal if it is within the transmission beam of the sender and if the beam falls at an angle less than the Field of View of the receiver. Assuming narrow beams, these links effectively work as point-to-point links working between two nodes exclusively. By placing a transmitter and a receiver hardware together and separate processing circuitry for both, these links can be made to work as bidirectional full-duplex links between two nodes and by having multiple instances of such configuration, nodes can have multiple bidirectional links [38]. However, in the latter case it is important to keep enough distance among the circuitries for different links to prevent interference. Also, the circuitry for each additional link incurs extra cost. Thus, it is important to keep the number of links for each node as small as possible. In our design, we do not allow a node to have more than three bidirectional links (except for the sink) and keep the spacing between two adjacent links of a node at least 90 degrees.

Note that we can use either one optical transceiver with steering hardware to rotate the device to cover different directions or multiple independent transceivers each dedicated to the communication to/from a particular direction [49, 13, 38]. In order to avoid the energy consumption due to steering and the need for line of sight tuning every time a switch occurs and to support parallelism and shorter delays, we use the latter model in our thesis where we have dedicated optical transceivers for each desired direction of communication. This model incurs higher deployment cost and thus calls for a well designed deployment scheme that minimizes the number of transceivers while maintaining desired properties. We focus on this deployment design problem in our thesis.

We consider a single static sink in the network instead of a moving AUV since collecting data using a moving AUV incurs higher delay of delivery because of the slow speed of the AUV compared the speed of optical communication. Also, a continuously moving AUV causes high energy consumption due to its movement.

As an example application of a high-speed and real-time optical underwater sensor network with static sink, we consider the VENUS project at the University of Victoria [46] deployed for seafloor observation from the shore. In this project, a sink (called a "node" in the VENUS project) is placed on the seafloor and it is connected to the shore using a fibre optic cable. Connected to this sink with wires are different "scientific instruments" or sensor nodes that transfer sensed data, *e.g.*, temperature, pressure, current, sound, image and video, to the sink in real-time and the sink in turn transmits data gathered from all the instruments to the shore using the fibre cable, also in real-time. The sink also supplies power to the sensor nodes. Connecting the sensors to the sink with wires limits the number of sensors that can be connected and introduces the overhead of wiring. In contrast, a network of sensors that communicates wirelessly and at the same time provides the bandwidth and propagation speed similar to a wired network would introduce ease of deployment, extendibility and greater area of observation. One of the five goals of the VENUS project is "High-speed and real-time connection to instruments" from the shore. With an optical wireless network of sensors around the sink, this goal is achieved with greater flexibility and extendibility. However, another goal of the VENUS project is "Unlimited power availability" which is currently ensured by directly supplying power to the sensors from the sink. With a wireless network of sensors around the sink, this becomes impossible. Still, we can deploy a hybrid network with some sensor nodes connected with wires to the sink to achieve unlimited power while some sensor nodes form a wireless optical network around the sink or around the nodes that are connected to the sink with wires to provide greater flexibility, extendibility and wider area of observation. Also, with the rapid development of power-harvesting technologies, it could be possible in future to design sensor nodes that harvest energy from the underwater environment by utilizing thermal vibration or kinetic energy of water.

2.3 Sensor Network Deployment

In our thesis, we consider the problem of deploying an underwater sensor network to achieve desired levels of connectivity and robustness while incurring minimum deployment cost. The problem of deployment to achieve a desired degree of coverage and/or connectivity has been investigated thoroughly in the literature of terrestrial radio-based sensor networks [23, 51, 2, 52, 61]. However, these works differ fundamentally from our thesis since they consider circular sensing region and circular communication region for sensor nodes to reflect omni-directional sensing and communication range. For underwater optical nodes, we have directional point to point links that can be modeled as straight lines with a limitation on the length. In our thesis, we assume that sensing ranges are much longer than communication ranges and therefore we only consider connectivity and ignore coverage in our work. Even if the sensing ranges are shorter, we believe that placing more nodes to cover all points in the target area is infeasible because of the high cost of sensor nodes. Instead, having a sparse but even distribution of sensor nodes throughout the target area should be sufficient for many applications.

Although not directly related to sensor networks, Kershner *et al.* [23] present analytical results that give important guidelines on the deployment of sensors with circular sensing region in a target field to cover the entire field with minimum overlap (and thus, minimum number of sensors). They prove that an optimal deployment of circles (requiring minimum number of circles) on a convex two-dimensional region is to place the circles (their centers) on the vertexes of an equilateral triangular lattice of side $r\sqrt{3}$ where r is the radius of each circle. Thus, if sensors have circular sensing region of radius R_s, we can place sensors on such lattice of side $R_s\sqrt{3}$ to achieve full coverage with minimum number of nodes. If communication range R_c is greater than or equal to $R_s\sqrt{3}$, then this deployment also ensures connectivity. For cases with $R_c < R_s\sqrt{3}$, Wang *et al.* [51] propose a strip-based deployment scheme where nodes are placed on horizontal strips with the spacing between the nodes and the spacing between the strips selected carefully and additional nodes placed between adjacent strips to connect them. Bai *et al.* [2] analytically prove that the strip-based scheme is asymptotically optimal (requires minimum number of nodes) for achieving coverage and connectivity when $R_c < R_s\sqrt{3}$.

Note that in addition to finding a deployment that minimizes the total cost, *e.g.*, number of nodes (and links for optical networks), it is also important to find a regular pattern that makes the task of deployment easier. For example, Bai *et al.* [2] suggest that using a complex pattern like the strip-based one can make the task of deployment difficult. Therefore, they discuss some simple deployment patterns like square pattern, rhombus pattern and hexagonal pattern and present the percentage of extra nodes introduced in the network resulting from the use of these non-optimal but regular patterns.

Pompili *et al.* [35] discuss the issues regarding the deployment of sensor nodes for an underwater acoustic sensor network. Since acoustic communication supports a very long range, the primary issue to deal with such cases is the sensing coverage. Pompili *et al.* [35] assume circular sensing region which makes the deployment task similar to that with terrestrial sensor networks, *i.e.*, the optimal way to deploy nodes in a two dimensional target region to ensure full coverage is to place nodes in a equilateral triangular lattice with sides $r\sqrt{3}$ where r is the sensing range. They formulate the trajectory of a sensor node as it sinks from the sea surface, where it is deployed, to the sea bottom given its initial velocity, velocity of the ocean current, buoyant and liquid forces and the dimension of the node.

Pompili *et al.* [35] also consider three-dimensional deployment and propose three schemes that can be used to achieve such deployment. The first and the simplest scheme is called 3D random where nodes are deployed at random on the sea-floor and then each node selects a depth at random and elevates itself to that depth by adjusting the length of the wire that connects it to the anchor. The second scheme in terms of simplicity is called Bottom-Random where nodes are deployed at random on the sea-floor and then a base station calculates the height of each node to achieve a desired level of coverage and each node elevates itself to that height. The third and the most complex scheme is called bottom-grid where nodes are placed on a square grid on the sea-floor with the help of one or more AUVs. Each sensor is assigned a height from the sea-floor in order to achieve a desired level of coverage and each node elevates itself to that height. Simulation shows that the bottom-grid scheme supports maximum level of coverage with minimum redundancy of all the three schemes. Note that with long range omni-directional acoustic media, connectivity is inherent and coverage becomes the principal goal for the deployment of acoustic underwater sensors. With short range directional optical media, connectivity becomes the principal issue for node deployment which is the focus of our thesis.

In contrast to carefully deploying sensor nodes to achieve desired coverage and/or connectivity with minimum number of nodes, extensive research has been conducted on the problem of deploying terrestrial sensor nodes randomly in large and redundant numbers and then selecting a subset of deployed nodes to achieve desired levels of coverage [60, 47, 42] or connectivity [59, 56, 55, 54, 3] or both [52, 61, 14]. These schemes are called node scheduling schemes and are designed to improve network lifetime by selecting and rotating a subset of deployed nodes to perform the sensing and communication tasks at a particular moment. With underwater sensor nodes, deploying nodes in redundant numbers is often not cost effective because of the high cost of sensor nodes [1]. In addition, with optical point to point links with limited range, placing the nodes carefully to ensure desired level of connectivity becomes the most important design goal.

2.4 Chapter Summary

In this chapter, we have presented the architecture of underwater acoustic networks and different protocols used at the MAC and network layer of such networks. We have seen that reliability and resiliency remain the most important design goals in the network layer of such networks due to the lossy nature of underwater acoustic media. We have demonstrated that extremely low bandwidth and long propagation delay make acoustic communication unsuitable for applications with stringent bandwidth and delay requirements. Then we have discussed research work and experiments on underwater sensor networks that use optical communication to achieve high bandwidth and fast propagation. We have also discussed physical characteristics and communication model for underwater optical systems with LEDs and photodiodes. Finally, we have discussed different network deployment schemes for terrestrial radio and underwater acoustic sensor networks and demonstrated how the deployment of underwater optical sensor nodes differs from the deployment of terrestrial radio or underwater acoustic nodes.

Chapter 3

Robust Grid-based Deployment Topologies

In this chapter, we design grid-based deployment topologies for underwater optical sensor networks. We assume a grid-based deployment where sensor nodes are placed on grid corners and then propose edge-selection patterns that allow us to select a subset of edges from the entire grid graph to achieve the desired levels of connectivity and path quality. We propose deployment patterns for cases where no node in the grid is allowed to have more than 1, 2 and 3 optical interfaces. We also evaluate the robustness and path quality of our proposed deployment patterns by simulating them using isolated and patterned failure models.

3.1 Problem Definition

We consider deploying sensor nodes in a two-dimensional gird lattice as shown in Figure 3.1. The grid consists of x columns and y rows, where $x = 12$ and $y = 12$ in the figure. All the unit squares have a side of length r where r is the maximum communication range of the optical communication system. In this thesis, we consider underwater optical communication using LEDs and photodiodes operating at green/blue visible light as transmitters and receivers, respectively. Additional concentrator lenses and amplifiers can be used with LEDs and photodiodes, respectively, to improve the range of communication. Such a communication system can have a value of r ranging from 8 meters to 40 meters depending on the specific communication devices used and the clarity of water [49, 13]. However, our design is independent of the actual value of r and is thus applicable to any configuration. Note that it is necessary to use narrow light beams with precise pointing if we want to support long range of communication. If the physical characteristics of the sensor nodes and the target underwater environment do not support this, we can use wider beams with shorter range but higher degrees of reliability and tolerance [49, 13].

We assume that each grid point has one sensor node placed on it, as shown if Figure 3.1. A gray line between two adjacent grid points in Figure 3.1 indicates a *potential* point-to-point optical communication link. That is, we can select *actual* links from this set of *potential* links to form a connected topology for the deployed nodes. The sink can be either one of the grid nodes or it can be a separate node placed within the vicinity of the grid area. Assuming that a node is placed on each grid point in order to maintain proper sensing coverage, having the length of each potential link equal to the maximum range r ensures that the separation between adjacent nodes is maximized and the total number of nodes (grid points) is minimized. As can be seen from Figure 3.1, a node has 4 potential links unless it is on the network boundary in which case it has 2 potential links if it is one of the four network corners and 3 potential links otherwise.

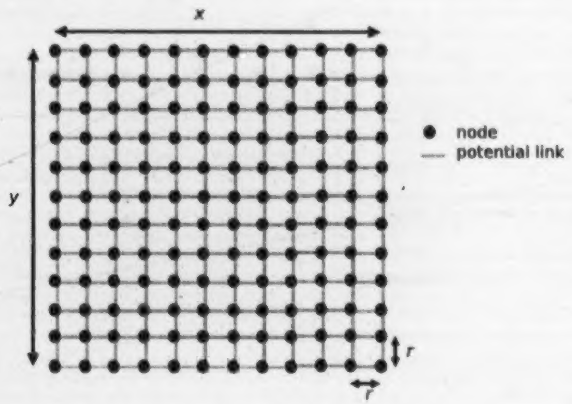

Figure 3.1: Grid-based deployment problem

The problem considered in this thesis is to select a set of *actual* links from the available pool of potential links as shown in Figure 3.1 so that a connected topology spanning all the sensor nodes and the sink is generated and the topology has the following three properties:

1) Robustness: The topology should have redundant links so that all or the maximum possible nodes are connected to the sink when one or more links are down. We primarily focus on link failures in our design since these are the main sources of failures in optical communication which relies on the line of sight property that can be obscured by underwater obstacles like underwater organisms, floating objects and sediments. We consider two types of robustness:

a) Deterministic Robustness: The topology should be 2-edge-connected [53], *i.e.*, the topology should be connected even if we delete an arbitrary link from it. We do not consider having 3 or more edge-connected topologies since the resulting topology would have excessively large number of links selected for it if we want 3 or more edge-connectivity. Besides, it is not possible to generate a 3 or more edge-connected topology from the potential grid of Figure 3.1 since this potential grid itself is not 3-edge-connected (we can disconnect any of the four corner nodes by deleting the only two links it has).

b) Probabilistic Robustness: The topology should be such that if we delete an arbitrary number of arbitrary links from it, a maximum possible percentage of nodes in the grid are still connected to the sink.

2) Path-quality: The topology should have minimum cost paths for each node to and from the sink in terms of number of links/hops that need to be traversed. Since the network is expected to operate most of the time without link failures, it is important to have minimum-cost paths in the topology from the sink to all nodes and vice versa in order to minimize the energy consumption resulting from communication. The alternate paths in the topology that will be used for communication in an event of one or more link failures should also be kept as short as possible in terms of number of links/hops.

3) Interface-count: Each link in the topology represents one communication interface/transceiver on the nodes at both ends of the link. Therefore, having more links in the topology to achieve the above two properties will introduce more interfaces per node and more total number of interfaces in the network and thus increase the total cost. Thus, we have to tradeoff property 1 and 2 above with per node and total interface count in the network. We can represent both the grid in Figure 3.1 and the selected topology as a graph by denoting the nodes and the sink as vertices and the links as edges. Thus, our goal is to select a subset of edges from all the potential edges of Figure 3.1 so that the robustness and path-quality of the resulting topology reaches a desired level and, at the same time, no node in the topology has a degree greater than a threshold and the total number of edges selected in the process is minimized.

In this thesis, we consider cases where each node in the topology is constrained to have no more than 1, 2 and 3 interfaces/degree since having more interfaces per node increases node cost and having more than 3 interfaces on one node is expensive in terms of cost and space. For each of the three cases, we develop patterns to select edges from the pool of potential edges to achieve desired levels of robustness and path quality while at the same time ensuring that a minimum total number of edges are added in this process. We consider placing one node at each grid point and then selecting the direction(s) in which a node can communicate so that when the network starts operating, good paths are available for low-cost communication and alternate paths are available if one or more links are down. The deployment task of placing nodes on grid points and setting their interface directions underwater can be done manually or using autonomous underwater vehicles (AUVs) [49].

It is important to note that we consider *deployment* in this chapter, not *operation*. In other words, we aim at finding a good initial setup of the network and the amount of resources (interfaces) needed for that setup. Network operation schemes for selecting paths, detecting faults and choosing alternate paths are considered in Chapter 4 where we perform dynamic evaluation of our deployment topologies. Also note that we do not consider having diagonal links in the grid since optical links have limited range and diagonal links are longer than horizontal and vertical grid links as shown in Figure 3.1. Also, having diagonal links would reduce the separation between adjacent links of a node which may introduce interference between the communications of the two adjacent links. With horizontal and vertical links as shown in Figure 3.1, a minimum separation of 90 degrees is maintained between any two adjacent links of a node.

3.1.1 Optical Interface Model

An optical interface consists of a transmitter and a receiver. It is cost-effective and intuitive to place both the transmitter and the receiver in one interface/board. In such case, a node can use one interface for transmitting to and receiving from the same direction. Thus we can model one interface as one undirected edge between two nodes. In other words, an undirected edge in our model indicates that each of the two nodes involved in this edge is using up one interface to send to and receive from the node at the other end of the edge. We use this model for the cases with maximum 2 interfaces per node and maximum 3 interfaces per node constraints as shown in Figure 3.2(b) and Figure 3.2(c). We call this the *undirected model* of interface.

For the maximum 1 interface per node case, this model would allow each node to have at most one undirected edge, thus making it impossible to generate a connected topology spanning all grid nodes. Therefore, for 1 interface per node case, we assume that each node can separate its receiver and transmitter portions of the interface since there is more room in the node now. Thus, each node can have at most one directed outgoing edge and one directing incoming edge and the incoming and outgoing edges need not originate from and point to, respectively, the same other node (see Figure 3.2(a)). We call this the *directed model* of interface.

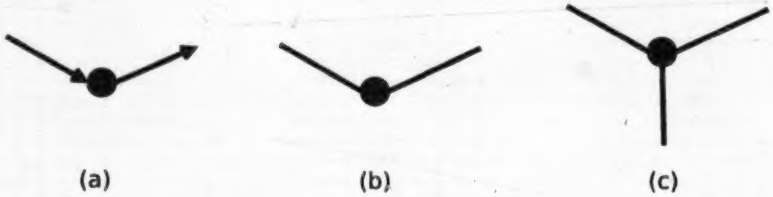

Figure 3.2: Optical interface model: (a) maximum one interface per node (b) maximum 2 interfaces per node (c) maximum 3 interfaces per node

3.2 Robust Deployment Schemes

In this section, we present schemes to select a set of edges from the available pool to generate robust deployment topologies for cases with maximum 1, 2 and 3 interfaces per node constraints. In particular, we come up with patterns that allow us to select the edges to produce deployment topologies with desirable properties.

3.2.1 Maximum 1 Interface per Node

As described in Section 3.1.1, having maximum one interface for each node means that each node can have at most one incoming and one outgoing directed edges. With such a configuration, the only way to ensure that each node is reachable both *to* and *from* the sink is to have cycles in the network. If we select a directed path from the sink to a node, we have to select a node-disjoint directed path from that node back to the sink in order to not violate the 1-interface constraint for the nodes. A Hamiltonian cycle starts from one node, visits all nodes in the graph exactly once and returns to the staring node [53]. By definition, all nodes are touched and because of the cycle property, each node has a directed path to each other node in the cycle. Each node is visited exactly once; therefore a node is entered once and left once which meets the one-interface constraint. Thus, if we assume that the sink is one of the grid nodes, a directed Hamiltonian cycle on the grid gives a topology with 1-interface constraint where all nodes are connected to and from the sink. A Hamiltonian cycle can always be formed in a grid graph if either the number of rows or columns or both are even [41]. Figure 3.3(a) shows this topology where the sink can be any grid node. The actual position of the sink does not make a difference in terms of path lengths in this topology since the topology is one big cycle covering all the grid nodes.

Although this topology satisfies the 1-interface constraint and provides the desired connectivity to and from the sink, it has poor robustness. Removal of just one edge in the cycle leaves one part of the network disconnected *from* the sink and the other part disconnected *to* the sink. Removal of edges closer to the sink leaves most of the network unreachable to/from the sink. Also, the path quality in this topology in terms of number of hops is poor on average because most nodes can actually be reached in a much shorter path from the sink than the paths available in the topology. However, with just one outgoing and one incoming edges per node, we can hardly do better than that.

A better design would be to divide the grid into four quadrants around the sink and form one directed Hamiltonian cycle for each quadrant. This is shown in Figure 3.3(b). In this topology, having the sink at the center of the grid minimizes the average length of the paths to and from other nodes. Therefore, we consider placing the sink only at the center with this topology. As shown in Figure 3.3(b), we have a sink node in this topology that is not a grid node. This has been done to keep the four Hamiltonian cycles non-overlapping so that they do not introduce nodes that violate the 1-degree constraint. Note in Figure 3.3(b) that the sink now has a total of eight directed edges. We assume that the sink can have more interfaces than the sensor nodes do, therefore this is not a violation of the 1-degree constraint. Also note that four of the eight edges of the sink are longer than the maximum range r. We assume that the sink uses powerful concentrator lenses and amplifiers to increase its range. Alternatively, we can introduce four additional nodes around the sink to reduce the length of communication.

The topology shown in Figure 3.3(b) has better path qualities than that in Figure 3.3(a). Also, it has better robustness since the removal of an edge disconnects only the nodes in the quadrant in which the edge is, nodes in all other quadrants are still connected to and from the sink. Better yet, all the edges in a quadrant can go down without disconnecting nodes in the other three quadrants. We just have to add some additional edges with the sink for all these advantages. Note that although the topology of Figure 3.3(b) has better robustness, it is still not 2-edge-connected, *i.e.*, we cannot remove an arbitrary edge from the topology without disconnecting it. We call this topology TOP1 in the rest of the thesis.

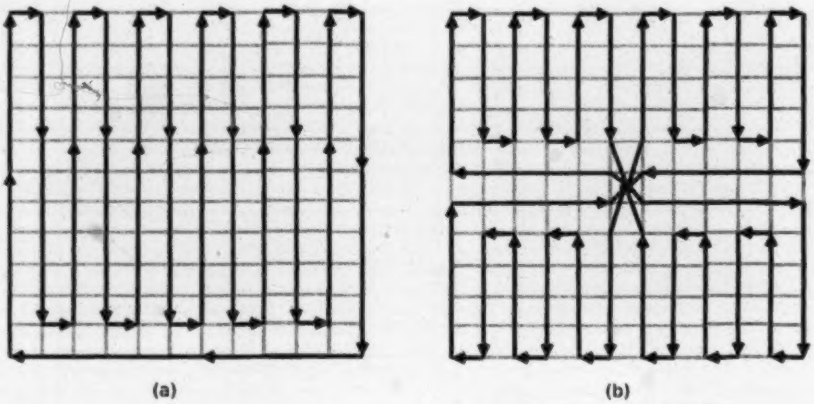

Figure 3.3: Topologies with 1-interface constraint: (a) One directed Hamiltonian Cycle (b) 4 directed Hamiltonian cycles (TOP1)

3.2.2 Maximum 2 Interfaces per Node

With maximum 2 interfaces for each node, we use the undirected interface model as described in Section 3.1.1. That is, we consider only undirected edges and allow each node to have a maximum degree of 2. With more freedom in the choice of interfaces, we consider improving the deterministic robustness of the topology. To this end, we build a 2-edge-connected topology so that any one edge can be removed from the topology without disconnecting it. The only way to build a 2-edge-connected topology with 2-degree constraint per node is to form a ring or cycle spanning all nodes [31]. Therefore, we can use the Hamiltonian cycle based topologies discussed in the previous section with directed edges replaced by undirected edges. Since four Hamiltonian cycles with a sink at the center of the grid (see Figure 3.3(b)) provides significant advantage in terms of path length and robustness, we consider only the topology with four undirected Hamiltonian cycles as shown in Figure 3.4. This topology is 2-edge-connected since we can remove an arbitrary edge from it without disconnecting the topology since we have two completely disjoint undirected paths from sink to each node in each cycle. Improving the probabilistic robustness of this topology requires adding some redundant edges to it. However, we cannot do this without violating the 2-degree constraint since each grid node already has a degree of 2. The path quality of this topology is still poor because most nodes are reached from the sink through a path that is considerably longer than the shortest path in the grid. We call this topology TOP2 in the rest of the thesis.

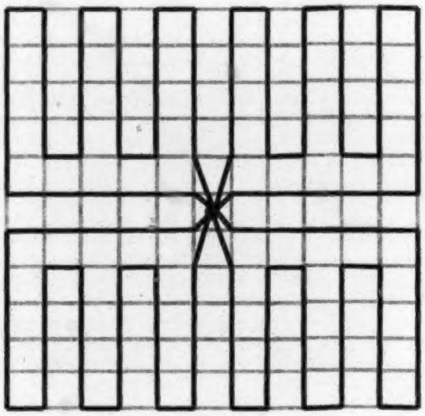

Figure 3.4: TOP2: 4 undirected Hamiltonian cycles

3.2.3 Maximum 3 Interfaces per Node

With the number of interfaces/degree each node can have increased to 3, we have enough freedom in our design to consider connected topologies that meet all the criteria described in Section 3.1, *e.g.*, deterministic robustness (2-edge-connectivity), probabilistic robustness (most of the nodes remain connected on the face of multiple edge removals) and short paths to and from the sink. Since the resulting topology must have shortest possible paths in the grid from the sink to each node in order to provide lowest-cost communication in a failure-free environment, our approach is to first build a 3-degree-constrained shortest path tree from the sink spanning all nodes and then add additional edges to this tree to first make it 2-edge-connected and then to further increase the redundancy of available paths to improve probabilistic redundancy. Since our final topology with 3-degree constraint will be 2-edge-connected, each node in the network will have at least a degree of 2. Therefore, our target is to minimize the number of 3-degree nodes as we move through each step. In the discussion below, we assume that the sink is one of the grid nodes.

Let us first form a shortest path tree from the sink to all nodes. Our goal is to select a set of edges from the grid to form a shortest path tree from the sink to all other nodes so that no node in the tree has a degree greater than 3 and the number of 3-degree nodes in the tree is minimized. We call such a tree an *optimal pattern* for the grid.

Definition 1. (Optimal Pattern) Given a grid graph and one of the grid nodes designated as the *sink*, an optimal pattern is defined as a 3-degree-constrained shortest path tree with minimum number of 3-degree nodes rooted at the sink and spanning all nodes in the grid.

Note that algorithms for generating a degree-constrained topology from a graph in general have been discussed in the graph theory literature [26, 24]. In our work, we find a deployment pattern that is 3-degree constrained, minimizes the number of 3-degree nodes and also produces shortest paths in a grid network of arbitrary size.

While forming the optimal pattern, we use the notion of *Manhattan Distance* [25] to find a shortest path from sink to a sensor node in the grid. Manhattan Distance between two points (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$ and is the length of the shortest path in terms of hops between the two points in a grid where a path can only have horizontal and vertical edges. Also, in a grid with only horizontal and vertical edges, any path from one node to another formed by never going back in a direction already used is a shortest path in terms of hops between the two nodes. Here, going back in a direction means going left in a path that already went right once and vice versa or going up in a path that already went down once and vice versa. This is shown in Figure 3.5 where both paths from S to D are shortest paths in the grid with a length of 13 hops since both paths are formed without ever going back in a direction. We call this the *Manhattan Distance Property*. This property implies that any shortest path between two nodes in a grid must be on or inside the rectangle formed by drawing vertical and horizontal lines through the two points, as shown in Figure 3.5.

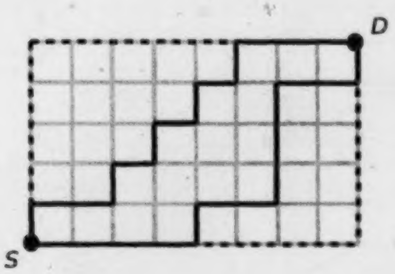

Figure 3.5: Manhattan Distance Property

For our grid, we call the horizontal and vertical lines through the sink the axes. The four quadrants created by the axes are called Q_1, Q_2, Q_3 and Q_4 starting from the upper left quadrant and traversing the quadrants clockwise around the sink (see Figure 3.6). According to the Manhattan Distance property, the only shortest path from the sink to a node on one of the axes is the path through the corresponding axis. Therefore, any shortest path tree spanning all nodes must include all the edges on the axes. As can be seen in Figure 3.6, The structures of the four quadrants are essentially the same. The upper-left (Q_1) quadrant can be described as a grid with a sink at the bottom-right corner and a horizontal and vertical axes through the sink on which all the edges are already selected for the target shortest path tree. Same description can be used for Q_2, Q_3 and Q_4 if we rotate them by angles of $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$, respectively, around the sink counterclockwise. Having four quadrants of the same structure, we now find the lower bound for the number of 3-degree nodes in a 3-degree constrained shortest path spanning tree from the sink for one such quadrant.

Theorem 1: Consider a quadrant with sides of size m and n where $m \leq n$. A sink is placed in a corner. A 3-degree constrained shortest-path tree rooted at the sink and spanning all nodes inside and on the boundaries of the quadrant requires at least $(m-2)$

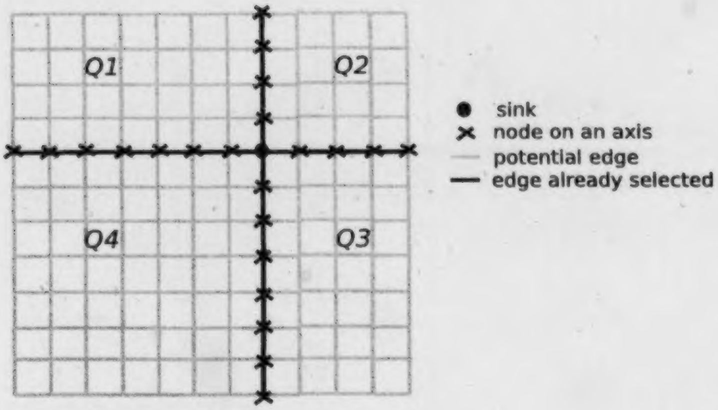

Figure 3.6: Four quadrants around the sink: edges on the axes cannot be avoided in a shortest path spanning tree from the sink

3-degree nodes.

Proof: Two axes extend from the sink. Consider the interior nodes of a diagonal from the top (furthest from sink) of the longest axis (the one with size n) to the edge of the quadrant. For example, this diagonal is shown as hollow circles in Figure 3.7(a) where $m = 7$ and $n = 8$. There are $(m - 2)$ nodes on the interior of this diagonal (that is, there are m nodes on the diagonal and we exclude the axis node and edge node). We will show that these $(m - 2)$ nodes, call this set of nodes D, require unique 3-degree nodes in their path to the sink.

Suppose we find a shortest path from one node a in D to the sink. According to the Manhattan Distance property, any shortest path between the sink and one of the $(m - 2)$ nodes must be within or on the rectangle created by extending straight lines from the node to the axes. For example, in Figure 3.7(b) the rectangle for a is shaded. Furthermore, the axis nodes that are at the ends of this rectangle must be interior nodes since the nodes of D are interior nodes.

To find a shortest path to the sink for this arbitrary interior diagonal node a, two cases can occur. In the first case, the selected path joins a path already created for another node in D. At the first node that it encounters this existing path, it will cause another edge to be added to this node. Note that no other nodes in D are within or on the rectangle of a. Therefore any paths already set up for other nodes of D will create 2-degree or 3-

degree nodes within the rectangle (no single degree nodes). Adding an edge to a 2-degree node creates a 3-degree node; adding an edge to a 3-degree node will violate the 3-degree constraint and is not allowed. The other case is that the path reaches an axis without encountering another path created by other nodes in D. All the edges on the axes have already been selected since these edges cannot be avoided to provide shortest paths to the nodes on the axes. Since the rectangle of a hits an interior axis point/node, this node must be a 2-degree node. Therefore, this also creates a 3-degree node. Both of these cases require the creation of a 3-degree node. Since this is true for an arbitrary node in D, it is true of all $(m-2)$ nodes in D resulting in at least $(m-2)$ 3-degree nodes. □

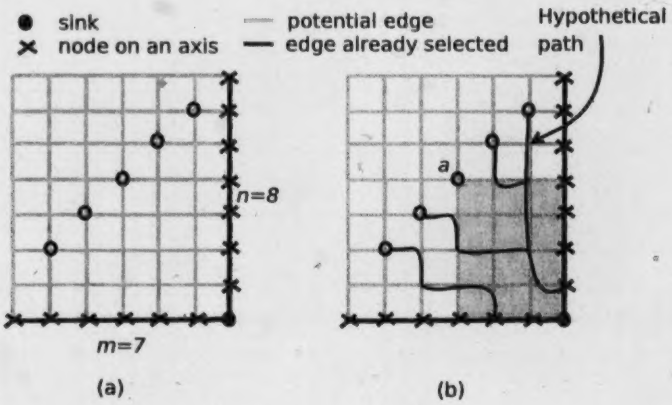

Figure 3.7: A quadrant with horizontal axis of length $m = 7$ and vertical axis of length $n = 8$

We now propose an edge selection pattern for a quadrant that results in a 3-degree constrained shortest path spanning tree from the sink with the minimum number of 3-degree nodes. We assume that all the edges on the axes are already selected. If the number of columns x (length of the horizontal axis) is less than the number of rows y (length of the vertical axis), then for each node on the horizontal axis we select all the edges that are on the straight line perpendicular to the horizontal axis at that point. All the nodes on the horizontal axis except the sink and the node at the other end of the axis have a degree of 3. Thus, the total number of 3-degree nodes is $(x-2)$. This is shown in Figure 3.8(a). If $y < x$, we follow the same pattern except that now we select all the edges perpendicular to the vertical axis and the number of 3-degree nodes is $(y-2)$. This is shown in Figure 3.8(b). If $x = y$, we can follow either of the above two patterns. Note in Figure 3.8(a) and Figure 3.8(b) that the path selected to each grid node is a shortest path from the sink according to the Manhattan distance property since these paths are formed by never going back in a direction already traversed. Also, no node has a degree greater than 3, thus the pattern indeed produces a 3-degree constrained shortest path spanning tree for the quadrant. In addition, according to Theorem 1, it has the fewest number of 3-degree nodes. Therefore, our pattern is an optimal pattern.

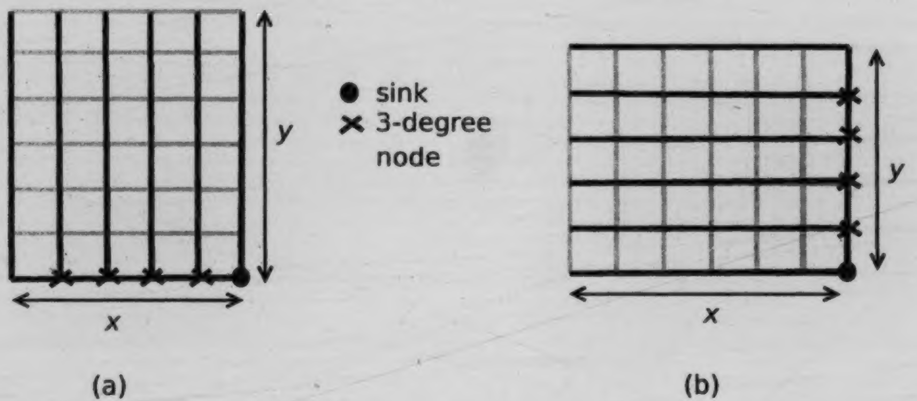

Figure 3.8: Optimal pattern for a quadrant: (a) $x < y$: total 3-degree nodes = $(x-2)$ (b) $y < x$: total 3-degree nodes = $(y-2)$

This gives us an optimal pattern for a single quadrant. If the sink is in the middle of the grid and the quadrants are all of the same size, then we can apply this to each quadrant as in Figure 3.9(a). According to the Manhattan Distance property, shortest paths to all nodes within or on the boundaries of a quadrant must remain on or within that quadrant, even when we consider the entire grid. Thus, an optimal pattern computed locally for a quadrant remains an optimal pattern for that quadrant when the entire grid is considered, provided that the 3-degree constraint is not violated. If the number of 3-degree nodes in the local optimal patterns for the four quadrants are l_i, $1 \leq i \leq 4$, then their sum $\sum l_i$, $1 \leq i \leq 4$, represents the lower bound for the number of 3-degree nodes in the globally optimal pattern for the entire grid. We call this lower bound LB. The case in Figure 3.9(a) gives exactly this minimum. However, if the sink is not central causing the quadrants to have different sizes, then the smallest axes may be shared by two different quadrants. Using this design would violate the 3-degree constraint. This is demonstrated in Figure 3.9(b) where both Q_1 and Q_2 draw edges from their common axis to form optimal patterns. In such cases, we have to find alternative patterns for at least one of the quadrants.

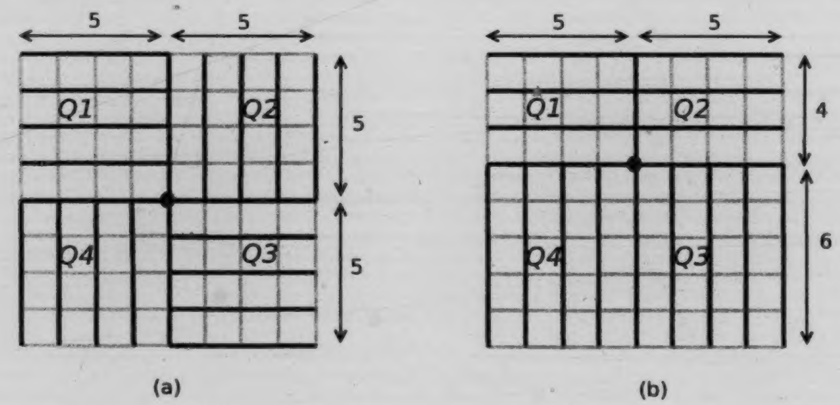

Figure 3.9: (a) Optimal patterns for individual quadrants lead to optimal pattern for the entire grid (b) Optimal patterns for individual quadrants violate the 3-degree constraint

To avoid violating the 3-degree constraint, we must ensure that each quadrant draws its edges from a different (unique) axis. For example, if each quadrant draws its edges from its clockwise right axis around the sink, there is no chance of conflict between two adjacent quadrants. Note that alternatively choosing the counterclockwise direction would result in the same property. The price to pay to achieve this conflict-free pattern for a quadrant is that the number of 3-degree nodes in the quadrant is “minimum” when the chosen axis is smaller than or equal to the other but “minimum+1” when the chosen axis is larger. The following corollary describes this pattern for the clockwise case (note that this is easily proved for the counterclockwise case as well).

Corollary: Consider a quadrant Q_i with a clockwise right axis of size y and a left axis of size x. Consider the pattern for Q_i shown in Figure 3.10 that draws edges only from the clockwise right axis (y). Figure 3.10 shows the pattern for different relationships between x and y. Let the number of 3-degree nodes in an optimal pattern for Q_i be l_i. The value of l_i can be found using Theorem 1. The number of 3-degree nodes for the pattern in Figure 3.10

applied on Q_i is l_i if $y \leq x$ but $(l_i + 1)$ if $y > x$.

Proof: For any x and y, the pattern is a shortest path tree (according to Manhattan Distance property) and does not have a node with degree greater than 3. For $y \leq x$, the number of 3-degree nodes is $(y - 2)$, as can be seen in Figure 3.10(a) and Figure 3.10(b). According to Theorem 1, this is the number of 3-degree nodes l_i in an optimal pattern for a quadrant Q_i with $y \leq x$. For $y > x$, the number of 3-degree nodes is $(x - 1)$, as can be seen in Figure 3.10(c) and Figure 3.10(d). According to Theorem 1, the number of 3-degree nodes in an optimal pattern for a quadrant Q_i with $y > x$ is $l_i = (x-2)$. Thus, the proposed pattern has $l_i + 1$ number of 3-degree nodes in Q_i when $y > x$. □

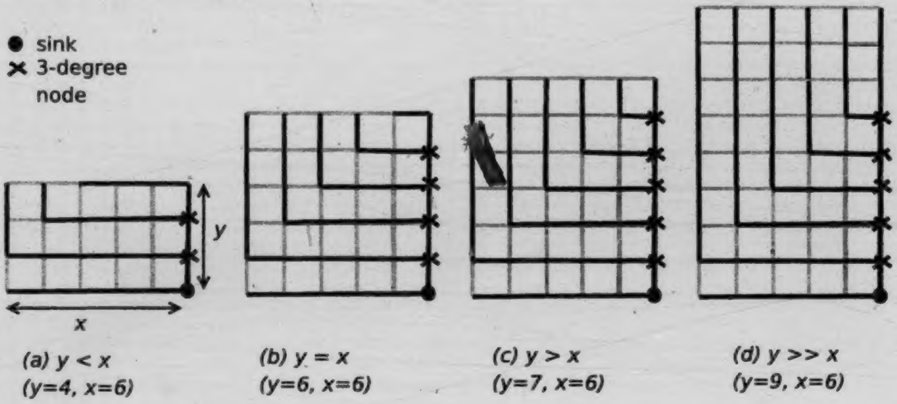

Figure 3.10: Pattern using only the vertical axis (clockwise right axis) to place 3-degree nodes

Now consider a grid with an arbitrary dimension and the sink placed at an arbitrary grid point. If we apply the pattern described in the corollary to individual quadrants of the grid, the overall grid is guaranteed to have nodes of at most degree 3 since each quadrant avoids conflict by placing its 3-degree nodes on its clockwise right axis. Thus, the resulting overall pattern, as shown in Figure 3.11(a), produces a 3-degree constrained shortest path tree from the sink to all grid nodes. We call the pattern in Figure 3.11(a) TOP3 in the rest of the thesis.

For a given sink placement, a pattern can be designed with at most (LB+2) 3-degree nodes where LB $= \sum l_i$, $1 \leq i \leq 4$. This can be seen by the following. In general, a pattern created by a clockwise choice of axis will have at most 3 quadrants forced to draw edges from their larger axes, creating (LB+3) 3-degree nodes. However, if this occurs, we could instead choose axes using the counter clockwise direction. In this case, the three quadrants will draw edges from their smaller axes but the fourth quadrant will draw edges from its larger axis. Thus, the number of 3-degree nodes will be (LB+1). There are also sink placements that create (LB+2) 3-degree nodes as shown in Figure 3.11(a). Therefore, for any sink placement, we can generate a pattern for the entire grid with at most (LB+2) 3-degree nodes using the corollary or its counterclockwise version.

TOP3 (see Figure 3.11(a)) ensures the best paths from sink to each grid node (and vice versa) with (LB+2) 3-degree nodes in the worst case. In addition to requiring only a small number of 3-degree nodes compared to all grid nodes, TOP3 offers nice regularity by placing all the expensive nodes (3-degree nodes) on the grid axes. Such regularity is desirable in node deployment since it often introduces convenience in the task of deployment [2]. However, TOP3 is a tree and hence has a very low level of robustness since there is exactly one path from sink to each node and vice versa. We now add additional edges to TOP3 to first make it 2-edge-connected to have the desired deterministic robustness and then to improve its probabilistic robustness. Our first step would be to connect the leaves together since that gives us the highest improvement in robustness. We note in Figure 3.11(a) that all the leaves of a quadrant are on the boundary of the quadrant and they are adjacent to each other on the boundary. Thus, we can connect all the leaves of a quadrant as shown in Figure 3.11(b). Since all the leaves are on the network boundary, adding such edges will not violate the 3-degree constraint. We apply this strategy in each quadrant which leads us to our next topology called TOP4 as shown in Figure 3.11(b). Note that TOP4 has considerably greater robustness than TOP3 since we can now remove any edge from the network, except the 4 edges of the sink, without disconnecting any node from the sink. Thus, TOP4 has high robustness but it is not 2-edge-connected.

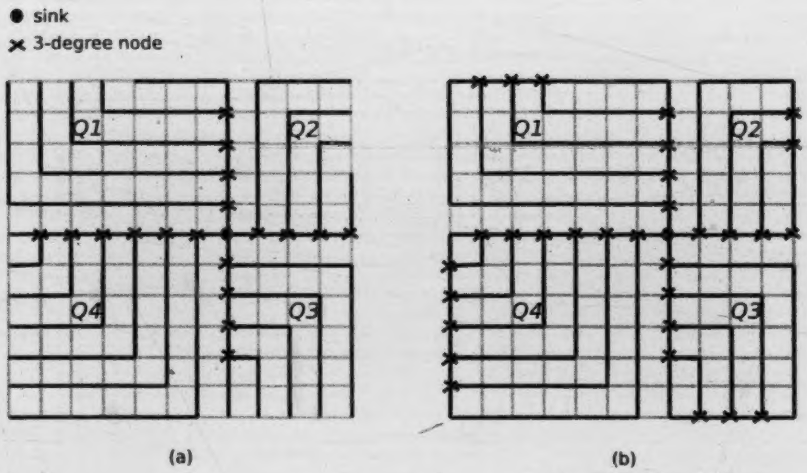

Figure 3.11: (a) TOP3: pattern from the corollary applied on 4 quadrants (b) TOP4: Leaves are connected

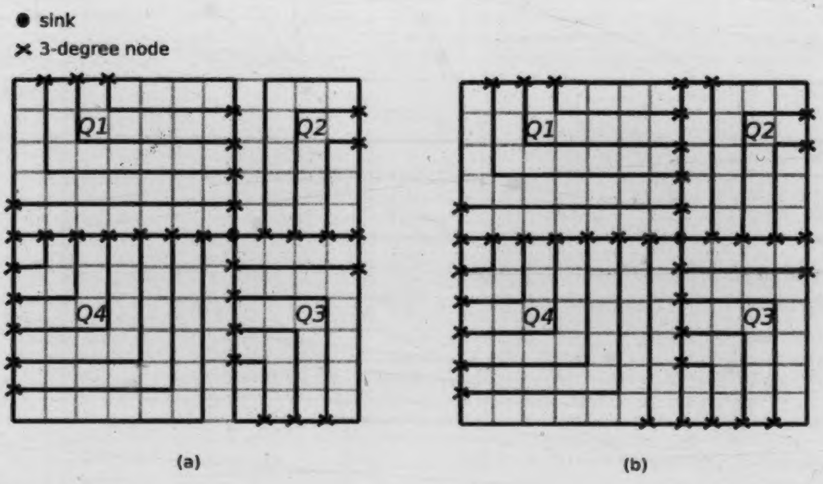

Figure 3.12: (a) TOP5: Q_1&Q_4 and Q_2&Q_3 are connected (b) TOP6: Q_1&Q_2 and Q_3&Q_4 are connected

We next add two more edges to TOP4 to make it 2-edge-connected and call it TOP5. We connect Q_1 and Q_4 with one boundary edge and Q_2 and Q_3 with another, as shown in Figure 3.12(a). TOP5 introduces 4 more 3-degree nodes on TOP4. The edge between Q_1 and Q_4 allows nodes in Q_1 to be reached through Q_4 and vice versa. Same is true for Q_2 and Q_3. Also note that TOP5 is 2-edge-connected since we can remove an arbitrary edge from it, even one of the four the sink edges, without disconnecting any node from the sink.

Our last topology TOP6 arises from the observation that all possible paths in TOP5 from the nodes in Q_1 and Q_4 to the sink are confined within Q_1 and Q_4. Similar for Q_2 and Q_3. If we add two more edges to TOP5, one to connect Q_1 and Q_2 and the other to connect Q_3 and Q_4, the number of possible paths from each node to the sink increases greatly. This new topology is called TOP6 and is shown in Figure 3.12(b). Now the path from a node in a quadrant can go through any of the other quadrants. This improvement in robustness introduces just two more edges and four more 3-degree nodes in the network. Note that TOP4, TOP5 and TOP6 all retain the regularity exhibited by TOP3 in the sense that all the expensive nodes (3-degree nodes) are placed on the grid axes and the network boundaries, not on arbitrary grid points.

We have thus designed a pattern for a deployment topology that is 2-edge-connected, has a significant number of alternate paths from each node to the sink and has shortest paths from sink to each node. While building this topology, we have introduced the least number of 3-degree nodes in the network at each step while all the nodes have at least a degree of 2 to ensure 2-edge-connectivity. We do not consider adding more edges to TOP6 since we believe that it is not possible to improve the robustness significantly by adding few more edges to TOP6. In the next section we show that TOP6 shows a high degree of robustness to isolated and patterned link failure models.

3.3 Static Evaluation of Proposed Deployment Topologies

In the previous section, we have designed six topologies for grid-based deployment for different constraints on the number of interfaces a node can have. These topologies are called TOP1, TOP2, TOP3, TOP4, TOP5 and TOP6 and have been shown in Figure 3.3(b), Figure 3.4, Figure 3.11(a), Figure 3.11(b), Figure 3.12(a) and Figure 3.12(b), respectively. These topologies have been summarized in Table 3.1. Various properties of these topologies and the entire potential grid graph when applied on a 12x12 grid with the sink placed at the center have been shown in Table 3.2. As can be seen from Table 3.2, TOP6 has a total of 165 edges/links whereas the potential grid has a total of 264 edges/links. This demonstrates the enormous savings in deployment cost that our proposed deployment topologies offer. As we shall see in this section, TOP6 offers a very high degree of robustness to isolated and patterned link failures inside the grid even though it has only a fraction of the links compared to the entire grid graph.

Topology	**Description**	**2-edge-connected?**
TOP1	4 directed Hamiltonian cycles	No
TOP2	4 undirected Hamiltonian cycles	Yes
TOP3	Shortest path tree from sink with 3-degree constraint and with (LB+2) 3-degree nodes in the worst case	No
TOP4	TOP3 with leaves in each quadrant connected together by a path	No
TOP5	TOP4 with quadrant 1&4 and quadrant 2&3 connected with two additional edges	Yes
TOP6	TOP5 with quadrant 1&2 and quadrant 3&4 connected with two additional edges	Yes

Table 3.1: Summary of proposed deployment topologies

Topology	**1-d nodes**	**2-d nodes**	**3-d nodes**	**4-d nodes**	**Total edges**	**Avg. node degree**
TOP1	144	0	0	0	148 (dir)	1.00
TOP2	0	144	0	0	148	2.00
TOP3	22	103	18	0	143	1.97
TOP4	0	111	32	0	161	2.22
TOP5	0	107	36	0	163	2.25
TOP6	0	103	40	0	165	2.28
Grid Graph	0	4	40	99	264	3.66

Table 3.2: Properties of different topologies when applied on a 12x12 grid with the sink at center

In our design in the previous section, we have made the topologies 2-edge-connected whenever we have enough interfaces to do so in order to achieve deterministic robustness. In this section, we evaluate the probabilistic robustness of these topologies by simulating them with probabilistic failure models. We call our evaluation in this section "static" since we do not consider the dynamic aspects of the network such as variation of failure probabilities with time, movement of link-blocking obstacles with time, dynamic selection of routing paths to avoid obstacles and network operations like routing, fault-detection etc. that changes with time. Dynamic evaluation of our topologies have been performed in Chapter 4. In our simulation in this section, we use a 12x12 grid with unit distance $r =$ 20m.

Our metric for robustness is defined as the percentage of nodes that are reachable from the sink when one or more links are failed according to the failure model under consideration. To see the quality of the available paths in case of failures, we also calculate the average length of the best (shortest) available path from the sink to each reachable node in the grid. Note that we do not include the nodes that are not reachable from the sink in this calculation.

3.3.1 Failure Models

We primarily focus on link failures in our evaluation since these are the main sources of failures in optical communication which relies on the line of sight property. We use failure models that are similar to the models used by Ganesan *et al.* [12] with some small modifications. For isolated failure, we fail each link in the topology with probability p_i and calculate our metrics for robustness and path quality. We simulate 1000 such runs and take their average in order to have acceptable 95% confidence intervals. Confidence intervals have been shown in each plot presented in Section 3.3.2. This failure model represents uncorrelated link failures in different parts of the network due to multiple small floating objects in different parts of the network, temporary displacement of a node that cause the loss of line of sight etc. We assume that nodes are anchored with the sea floor to keep them in place.

For a patterned or correlated failure model, Ganesan *et al.* [12] fail all nodes within a randomly placed circular area. We use ellipses with small semi-minor axis b and large semi-major axis a, as shown in Figure 3.13, to model underwater floating objects and organisms. We call such an obstructing ellipse an *error blob* in the remainder of the thesis. In each run, we select k random locations inside the grid as the centers of k error blobs and we select a random orientation for each error blob. Here, k is a Poisson random variable with mean λ. We fail all links/edges that are completely or partially inside these error blobs and calculate our metrics for robustness and path quality. We simulate 1000 such runs and take their average to achieve acceptable 95% confidence intervals.

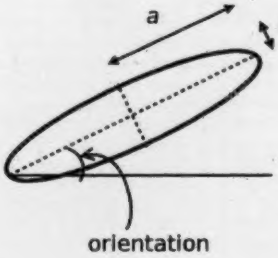

Figure 3.13: An "error blob": an ellipse representing an underwater obstacle

3.3.2 Results

We present the results with the sink placed at the center of the grid. For TOP1 and TOP2 both of which use four Hamiltonian cycles, the sink is a separate node placed at the geographic center of the grid area. For shortest-path based topologies (TOP3 to TOP6), the sink is the grid node that is closest to the geographic center of the grid.

Figure 3.14 presents the robustness of the six topologies to isolated failure. As expected, the directed Hamiltonian cycle based TOP1 that uses 1 interface per node exhibits the worst robustness. TOP1 uses four directed cycles to connect all the nodes to and from the sink. The removal of a single link on such a cycle causes one half of the cycle to be disconnected from the sink and the other half to be disconnected to the sink. This is why robustness of TOP1 rapidly goes down as soon as the failure rate p_i is increased above 0. As can be seen in Figure 3.14, the undirected Hamiltonian cycle based TOP2 and the shortest path tree based TOP3 exhibit close robustness. Since TOP2 uses undirected cycles, we can remove one arbitrary link from each of the four Hamiltonian cycles in TOP2 without disconnecting any node from the sink. Even if we remove multiple links from a cycle, the number of nodes that are disconnected from the sink is small as long as the removed links are geographically close. As a result, for smaller values of p_i, the robustness of TOP2 is fairly good and better than that of TOP3 which is a tree. As the failure rate increases beyond 5%, robustness of TOP2 falls below that of TOP3. This is because at higher p_i, the probability of simultaneous failures of two links that are far apart in the cycle increases. Such a failure effectively disconnects all the nodes between these two links in the cycle from the sink. The robustness of TOP3, which is a tree, falls less steeply than that of TOP2 at higher values of p_i. This is because removing multiple links from the tree disconnects only the descendants of each edge in the tree rather than disconnecting a large portion of the cycle as in TOP2. Once the leaves are connected together to generate TOP4, we see a significant improvement in the robustness. The robustness of TOP4 falls steadily with the increase of failure rate p_i and 80% of the nodes are connected to the sink even with a high failure rate of 8%. TOP5 and TOP6 show similar characteristics with small but consistent improvement in robustness at high failure rates. Note that TOP5 and TOP6 use just 2 and 4 extra links, respectively, on TOP4. Therefore, this improvement in robustness comes without a significant price.

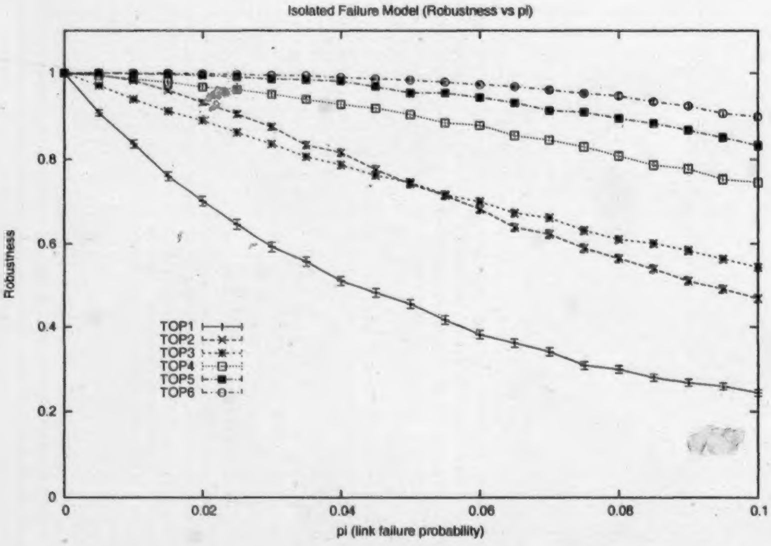

Figure 3.14: Robustness to isolated failure

Figure 3.15 shows the average path length of the connected nodes from the sink under isolated failure as a function of failure rate p_i. As expected, TOP1 and TOP2 have the longest path lengths. Initially, path length for TOP1 is twice the path length for TOP2 since TOP1 uses only directed edges. The path length of TOP1 decreases sharply with higher

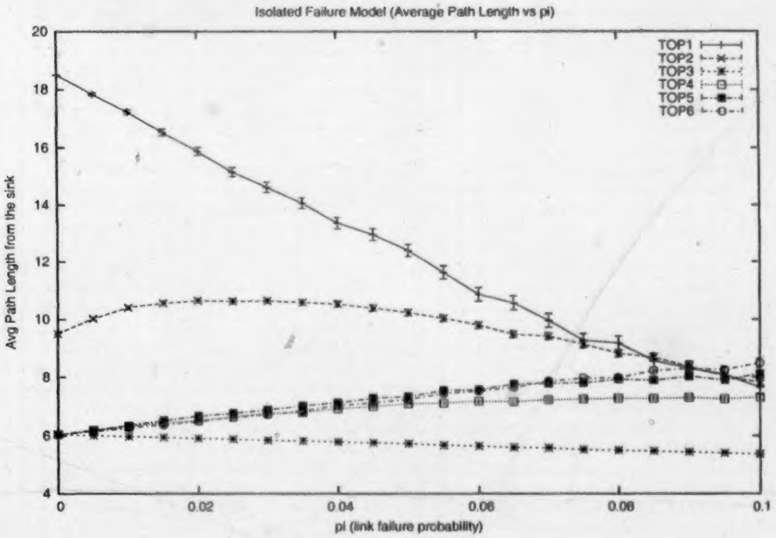

Figure 3.15: Average path length: isolated failure

failure rates. This is not because the paths are becoming better but because more nodes are becoming disconnected and we do not include disconnected nodes in our metric. For TOP2, average path length increases initially since a high level of robustness is maintained and longer paths are used to keep the nodes connected. As the failure rate goes up, the average path length for TOP2 begins to fall gradually since more nodes are becoming disconnected. TOP3 to TOP6 are all built upon a shortest path spanning tree. Thus, they all have optimal average path length at no-failure condition. The path length metric decreases very slowly for TOP3 (tree) with higher failure rates since more nodes are becoming disconnected. For TOP4, TOP5 and TOP6, path length metric slowly increases with higher failure rate. This is because in the presence of link failures, these topologies use alternate sub-optimal paths to the sink. This reflects the fact that under higher failure rates, traffic will get through, though on longer paths.

Figure 3.16 shows robustness of the proposed topologies to patterned failure with an elliptical failure model. The semi-minor axis b has been kept fixed at 4m and the semi-major axis a has been varied along the x-axis. We choose the values of a that results in a total area of the ellipses that we would get if we concentrate the isolated link failure probabilities into a smaller area so that each link in that area has a failure probability of 1. We choose the range of a to be from 7m to 90m which reflects isolated failure p_i from 0.005 to 0.06 in Figure 3.14. The value of λ has been kept fixed at 3. As can be seen if Figure 3.16, the relative performance of the six topologies is the same as that with isolated failure shown in Figure 3.14. However, the difference between the performances of the various topologies is significantly smaller in Figure 3.16. Also, the robustness of TOP1 falls gradually and the robustness of TOP2 remains higher than that of TOP3 (tree) all the time. This is because with patterned failure, links removed from Hamiltonian cycle based TOP1 and TOP2 are close together which causes a smaller part of the cycle(s) to be disconnected from the sink. The robustness of TOP4, TOP5 and TOP6 are consistently higher than that of TOP2 but the difference is not significant. This is an indication that if link failures are geographically close (patterned), TOP2 with undirected Hamiltonian cycles is a very good candidate for deployment.

The average path lengths under patterned failure model are shown in Figure 3.17. As in isolated model, TOP1 and TOP2 exhibit the worst path lengths, with TOP1 showing almost twice the path length of TOP2. However, unlike the isolated model, the path length of TOP1 decreases gradually with larger sizes of the ellipses. This is due to the fact that in patterned model the number of nodes disconnected from the sink increases very slowly with higher ellipse sizes for TOP1 (see Figure 3.16). Similar behavior is observed for TOP3 (tree) with the exception that the average path length starts with the optimal. For TOP2, TOP4, TOP5 and TOP6, the average path length increases slightly to allow alternate sub-optimal paths for nodes. However, this increase is negligible compared to that in isolated failure (see Figure 3.15) because link failures concentrated in specific geographic regions destroy best paths of fewer nodes.

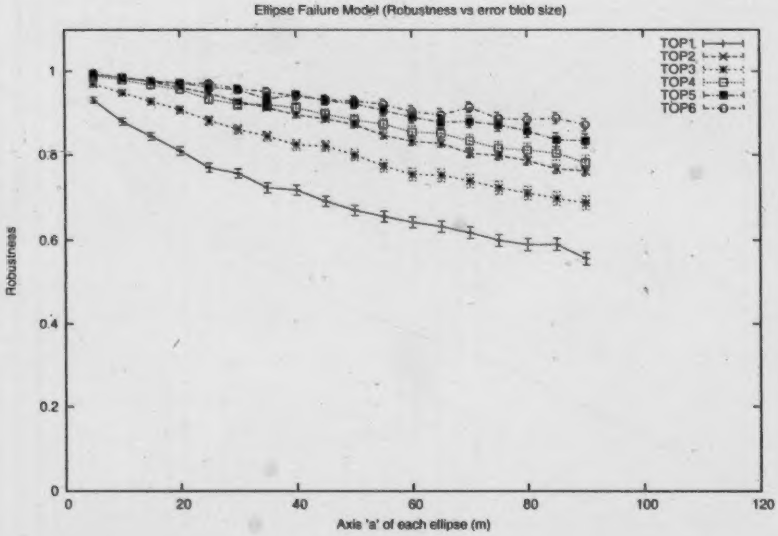

Figure 3.16: Robustness to patterned failure with an elliptical failure model. b = 4m and $\lambda = 3$

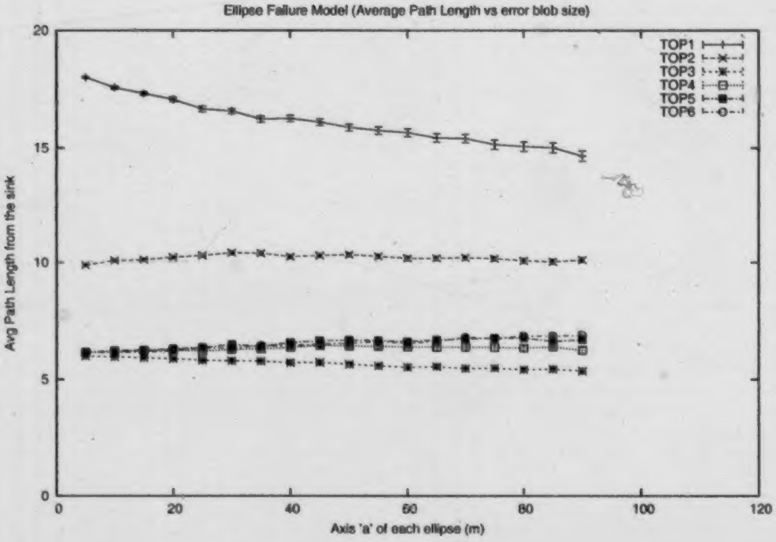

Figure 3.17: Average path length under elliptical failure model. b = 4m and $\lambda = 3$

We have also experimented with other sink placements in the grid. For example, with the sink at the corner, there is just one quadrant in the entire grid. Thus, the number of alternate paths for a node is decreased which causes lower robustness. The average path lengths increase since the sink is far from many nodes. Other than this, different topologies exhibit similar relative behavior. Therefore, we do not present our results with the sink placed at a grid corner. We have also experimented with node failures instead of link failures with isolated and patterned models. Robustness and path length with node failures exhibit similar behavior as with link failures except that the robustness of each topology is slightly lower with node failures than with link failures, especially at higher values of failure rates.

3.4 Chapter Summary

In this chapter, we have designed robust two dimensional grid-based deployment patterns for underwater optical sensor networks. We have formulated deployment topologies for cases where nodes are constrained to have no more than 1, 2 and 3 optical interfaces. For 2 and 3 interfaces per node constraints, we have proposed topologies that are 2-edge-connected to introduce deterministic robustness. For 3 interfaces per node constraint, we have designed robust topologies where each node can be reached using a shortest path from the sink. While designing these topologies, we have introduced least number of links in the grid in order to minimize the cost of deployment. To examine the probabilistic robustness, we have simulated our topologies using isolated and patterned failure models. Results show that our best topology TOP6 maintains a very high degree of robustness although it has only a fraction of the links present in the potential grid graph. Directed Hamiltonian cycle based TOP1 shows worst performance in terms of robustness and path quality. Because of its poor performance, we do not include TOP1 in our dynamic evaluation presented in the next chapter.

Chapter 4

Dynamic Evaluation of Deployment Topologies

In this chapter, we evaluate the dynamic behavior of our proposed deployment topologies by simulating three simple routing protocols on these topologies. Because of the presence of dedicated point to point optical links, nodes do not contend for channel access and this eliminates the need for a sophisticated Medium Access Control (MAC) layer. Therefore, our focus is on the network layer which is responsible for routing packets towards the sink.

Since optical links depend on the line of sight property, they can occasionally fail due to the presence of obstacles in the underwater environment such as underwater organisms, floating objects, sediments etc. Therefore, the routing schemes need to be resilient in the sense that they should be able to route packets around occasional underwater obstacles that obscure optical links and deliver packets by dynamically adjusting the routes to the sink. While doing so, the routing schemes should avoid introducing unreasonable delay and communication in the network that may degrade the quality of service and increase energy consumption. Note that the degree of resiliency that a given routing scheme can provide depends on the degree of redundancy that the underlying deployment topology possesses since higher degree of redundancy increases the probability that the routing scheme finds an alternate path to the sink when the original path is obscured by an obstacle. We use three simple routing schemes that handle the problem of resiliency in three different manners and consider the performance of these routing protocols on our topologies in terms of resiliency to link failures, average delay of delivery of packets to the sink and overall communication overhead.

4.1 Routing Protocols

We simulate three simple routing protocols on our topologies: a flooding protocol (FLD), a multi-path protocol (DPP) and an adaptive single path protocol (HHA). Our main goal is to evaluate and compare the proposed deployment topologies rather than formulating an optimal routing protocol. Therefore, while designing these protocols, we try to keep them simple and make them attack the routing and robustness/reliability problems from three different angles in order to produce a comprehensive evaluation of our topologies.

4.1.1 Flooding (FLD)

We use traditional memory-constrained flooding [45] as our first routing scheme. Here, a source node forwards its packet to all neighboring nodes. Each node receiving this packet forwards it to all its neighbors except the one from which the packet arrived. No acknowledgments are used: a best-effort delivery is assumed. Each node also remembers in its FLOODING_TABLE which packets it has seen so far. A packet is identified by the source and sequence number of the packet. If a node receives a packet that it has seen before, it simply discards the packet. On the other hand, if it receives a packet it has never seen before, it forwards the packet accordingly and inserts the packet's source and sequence number into the FLOODING_TABLE to avoid forwarding it again in the future.

Although flooding causes excessive transmission and is therefore an impractical scheme to use in a sensor network, we use flooding in our evaluation since it utilizes all possible paths from the source to the sink and thus shows us the degree of redundancy present in a topology and the maximum degree of robustness that we can expect from it. Also, flooding delivers a packet on the shortest available path at a particular moment and thus incurs the minimum delay of delivery. Thus, flooding gives us important insight about the topologies and their redundancies.

In FLD, each node needs two kinds of storage: storage for FLOODING_TABLE and storage for keeping packets temporarily if the target outgoing links are busy transmitting previously received packets. The FLOODING_TABLE can be very large and its size depends on the number of source nodes present in the network and the number of packets generated by each source. The length of the FLOODING_TABLE can be kept shorter by applying intelligent maintenance schemes. In our implementation, we assume infinite storage for both FLOODING_TABLE and packet buffer. In FLD, nodes do not need to know the topology of the entire network since flooding does not use topology information in its forwarding logic. Also, there is no initial setup phase needed for FLD.

4.1.2 Dual Paths Protocol (DPP)

Dual Paths Protocol (DPP) utilizes the principle of multi-path routing [48, 30, 50] where a packet is routed on more than one path in order to improve the probability of successful delivery. In DPP, each source node initially computes two paths to the sink and forwards each packet on both paths in order to achieve fault tolerance in the presence of link failures. At the start of network operation, each source node computes its two paths and informs all nodes that are on either of the paths of this information so that these nodes can do the forwarding task when they receive packets from that source.

With TOP2, TOP5 and TOP6 all of which are 2-edge-connected, DPP selects two completely disjoint shortest paths from each source to the sink. With TOP4, DPP selects two paths from each source that are completely disjoint except for the link and node that connects to the sink. With TOP3, there is only one path (shortest) from each source to the sink and DPP selects this single path for packet delivery. In this sense, DPP on TOP3 becomes a single path protocol rather than a dual path protocol. Like FLD, no acknowledgments are used and a best-effort delivery is assumed.

We use DPP as an intermediate scheme between FLD that floods the network with a packet to achieve robustness/reliability by utilizing the redundancy in the topology and HHA (described next) that uses a single but dynamically adjusted path for delivery to achieve the same goal. Note that unlike most multi-path routing schemes [48], DPP is static in the sense that it computes the paths initially and never recomputes them based on current network conditions. Because of this static nature, DPP fails to fully utilize the underlying redundancy like FLD and HHA and thus is expected to have lower robustness/reliability than a dynamic multi-path scheme. However, DPP gives us important insight on the average delay and average number of transmissions per packet that we can expect from a typical multi-path scheme [48, 30, 50] since it uses two "shortest" paths to the sink (whenever available).

In DPP, a node does not need a FLOODING_TABLE and the storage to maintain such a table. However, like FLD, a node in DPP does need storage to keep packets temporarily if the desired outgoing links are busy forwarding previously received packets. In addition, each node needs a FORWARDING_TABLE to store routing information, *e.g.*, which outgoing link a packet should be forwarded on if the packet is originating from a particular source node. The size of this table is $O(N)$ where N is the number of nodes in the network. The entries in this table are calculated during network setup. However, this setup phase does not incur any transmission given that each node knows the deployment topology of the network which takes up $O(L)$ storage space at each node where L is the number of links in the network.

4.1.3 Hop-by-Hop Acknowledgment with Local Update (HHA)

The Hop-by-Hop Acknowledgment with local update (HHA) protocol delivers all packets on a single path and dynamically adjusts the path when a packet cannot be forwarded on the original link. It makes use of acknowledgment packets (ACK) on each intermediate hop to determine whether the packet is successfully received by the next hop neighbor. If not, it recomputes a new remaining path from the current node to the sink and forwards the packet accordingly. If no path to the sink currently exists, the packet is kept in the queue and transmission is tried again after a RETRY_INTERVAL. This process continues until the packet is delivered to the sink or a time greater than a parameter TTL (time to live) has passed since the generation of the packet, whichever occurs first.

In order to keep the protocol simple, a source node includes the entire path (all the node IDs on the path in strict order) to the sink in the REMAINING_PATH field of the packet. Each intermediate node removes one node from the REMAINING_PATH field of the packet and forwards it accordingly. If the packet cannot be forwarded to the next neighbor as suggested by the REMAINING_PATH field of the packet, the intermediate node recomputes a new remaining path from itself to the sink, if any, replaces the old remaining path in the packet with this newly computed path and forwards the packet accordingly.

In order to prevent a recomputed path to include a link the failure of which caused the packet to reach the current node in the first place, each node maintains a data structure called LINK_STATUS that reflects which of its outgoing links are currently down and while forwarding a packet the node appends this information in the LINKS_DOWN field of the packet. If a node has to recompute a path, it does so by applying shortest path algorithm on the original topology after removing the links indicated by the LINKS_DOWN field of the packet in question and its current LINK_STATUS data structure. This mechanism ensures that when a node recomputes a path for a packet, it does not include in the path the links that the packet observed to be down on its way from the source to the current node. Note that this information in a packet's LINKS_DOWN field is not stored locally by the recomputing node, making it necessary to include this information in each subsequent packet by the preceding nodes. A more intelligent protocol will have the nodes store this information for some time and use it intelligently in order to reduce the amount of data transmission. This is left for future work.

Details of the various aspects of the HHA protocol are discussed below.

Data Structures

Each node has one first-in-first-out buffer called PACKET_BUFFER and whenever a packet is generated at this node or a packet arrives at this node from a neighbor, the packet is placed on its PACKET_BUFFER, unless the packet is destined for this node. Packets are tried from the head/front of PACKET_BUFFER and removed from PACKET_BUFFER only when they have been successfully forwarded or dropped because they are too old. Another important data structure of each node is the LINK_STATUS data structure that reflects which of the outgoing links from this node are currently down. Whenever a node, say node s, finds that its link to neighbor n_1 is down (did not receive acknowledgment), it marks link (s, n_1) in its LINK_STATUS to be down. When the link is restored again, this link is marked to be up in the LINK_STATUS. The restoration of a link can be detected by periodically sending echo packets or by having a timer the expiration of which roughly indicates the restoration or any other hardware assisted technique. Finally, each node initially computes a shortest path to the sink in the initial target topology (*e.g.*, TOP2 or TOP5), assuming all links are up, and stores this path in a data structure called PRIMARY_PATH.

Generation of a Packet at a Node

When a packet is generated at node s, node s places the whole primary path (as indicated by its PRIMARY_PATH data structure) in the REMAINING_PATH field of the packet and puts the packet on the PACKET_BUFFER. The LINKS_DOWN field of the packet is left empty. The procedure to transmit a packet from the head/front of PACKET_BUFFER is described below.

Forwarding a Packet from a Node

Step 1: When looking at the packet at the head/front of PACKET_BUFFER, node s first checks to see if the time since the packet was generated is greater than a threshold TTL (time to live). If so, the packet is dropped and the next one in PACKET_BUFFER is tried. If not, node s looks at the REMAINING_PATH field of the packet in question and finds out the next hop node (neighbor of s). Let this next neighbor be node n_1. Two cases may arise which are described in step 1(a) and 1(b) below.

Step 1(a): (The link from node s to node n_1 is currently up according to LINK_STATUS data structure of node s) Node s Removes n_1 from REMAINING_PATH field of the packet, appends the links that are currently down (according to LINK_STATUS of node s) to the LINKS_DOWN field of the packet, transmits the packet to n_1 and waits for an ACK packet from n_1 for a time ACK_INTERVAL by starting a timer of the same duration. If an ACK from n_1 arrives before this timer expires, the packet is removed from the head/front of PACKET_BUFFER and the next packet in the buffer is tried in the same way (go to step 1 above). If no ACK arrives within this time, node s marks link (s, n_1) to be down in its LINK_STATUS structure. If this makes all outgoing links from node s to be down, node s waits for a time interval RETRY_INTERVAL and starts over the whole process (go to step 1 above). If, on the other hand, there is at least one currently active/up outgoing link from node s according to its LINK_STATUS, node s performs the procedure described in step 2 below (re-computation).

Step 1(b): (The link from node s to node n_1 is NOT currently up according to the LINK_STATUS data structure of node s) If all outgoing links from node s are down according to its current LINK_STATUS, node s waits for a time interval RETRY_INTERVAL and starts over the whole process (go to step 1 above). If, on the other hand, there is at least one currently active/up outgoing link from node s according to its LINK_STATUS, node s performs the procedure described in step 2 below (re-computation).

Step 2: (Re-computation of remaining path) Node s recomputes a best path from s to the sink by applying a shortest path algorithm on the initial topology after removing all the links from the topology that are down according to the packet's LINKS_DOWN field and the current LINK_STATUS data structure of node s. Then node s replaces the REMAINING_PATH field of the packet with this newly computed remaining path, appends the links that are currently down (according to LINK_STATUS of node s) to the LINKS_DOWN field of the packet, forwards the packet to the neighbor, say n_2, as suggested by the new path and waits for an ACK packet from n_2 for a time ACK_INTERVAL by starting a timer of the same duration. If an ACK from n_2 arrives before this timer expires, the packet is removed from the head/front of PACKET_BUFFER and the next packet in the buffer is tried in the same way (go to step 1 above). If no ACK arrives within this time, node s marks link (s, n_2) to be down in its LINK_STATUS structure and goes back to step 1 above. We go back to step 1 instead of doing another re-computation because the original link intended for this packet may have come back up by this time or the packet could have become so old that it is worth dropping and the next packet in the buffer is to be transmitted.

An example of how HHA dynamically adjusts the routing path is shown in Figure 4.1. The node numbers shown in the figure works as node IDs. Node 6 wants to send a packet to the sink (node 67) and puts the packet on its PACKET_BUFFER with REMAINING_PATH field set to be identical to its PRIMARY_PATH data structure which is the shortest path from node 6 to the sink in the initial topology (TOP6 in the figure). In the figure, the path is 6-7-19-31-43-55-67. The forwarding process in node 6 forwards the packet to node 7 and gets an ACK from 7 before the timeout occurs. Thus, node 6 forgets about the packet and it becomes the responsibility of node 7. The packet travels in the same way to node 19 and 31. Node 31 forwards the packet to node 43 but does not get an ACK before the timeout since node 43 does not receive the packet because the link (31-43) is obscured by an obstacle (shown as an ellipse in Figure 4.1). Thus, node 31 marks link (31-43) to be down in its

LINKS_DOWN data structure and recomputes a new path from node 31 to the sink taking into account the fact that link (31-43) is now down. In Figure 4.1, this new path is 31-19-7-8-20-32-44-56-68-67. Node 31 now assigns this path to the REMAINING_PATH field of the packet, appends link (31-43) in the LINKS_DOWN field of the packet (which was empty so far) and forwards the packet to node 19. The packet travels through nodes 19, 7, 8, 20 and 32 without any problem. When node 32 forwards the packet to node 44, it does not receive and ACK since link (32-44) is also physically obscured. Thus, node 32 recomputes yet another remaining path from itself to the sink taking into account that links (31-43) and (32-44) both are down. This new path is 32-20-8-9-10-11-23-22-21-33-45-57-69-68-67. Node 44 now assigns this path to the REMAINING_PATH field of the packet, appends link (32-44) in the LINKS_DOWN field of the packet and forwards the packet to node 20. The packet travels this new path without any obstruction and reaches the sink without any more re-computation. Note that if node 31 did not append in the LINKS_DOWN field of the packet that link (31-43) was down, node 32 would have no idea about this failure and it would recompute a path 32-20-8-7-19-31-43-55-67 which would cause the packet to travel back and forth between node 31 and 32 until the obstacle is gone.

HHA is thus a single path scheme but the path is dynamically adjusted to route around an obstacle. It utilizes the underlying redundancy in the topology to achieve a high degree of resiliency. This may occasionally produce very long paths in the presence of obstacles and longer paths cause longer delays and more transmissions. However, assuming that obstacles are infrequent, mobile and transitory, packets will be mostly delivered on short paths and very infrequently on longer paths to route around obstacles. Therefore, we expect that delay characteristics of single-path based HHA would be comparable on average with that of multi-path based DPP and flooding and number of transmissions with HHA would be smaller on average compared to multi-path based DPP and flooding. The latter observation arises from the fact that even DPP always forwards a packet on two paths causing a large number of transmissions whereas HHA forwards a packet on a single path although the path can be very long occasionally. We present quantitative comparison of these protocols in Section 4.5.

In HHA, a node does not need a FLOODING_TABLE and the storage to maintain such a table. However, a node does need storage to maintain the PACKET_BUFFER. We assume infinite space for PACKET_BUFFER at each node in our implementation. Each node also needs to know the deployment topology which needs a storage space of $O(L)$ at each node where L is the number of links in the network. There are two other data structures that a node in HHA needs to maintain: LINK_STATUS and PRIMARY_PATH. LINK_STATUS needs a storage space of $O(1)$ since the maximum number of links a node can have is constant. PRIMARY_PATH needs a storage space proportional to the maximum length of a shortest path in hops from a source to the sink. The setup phase consists of computing PRIMARY_PATHs from each node to the sink which does not incur any transmission given that the nodes know the deployment topology.

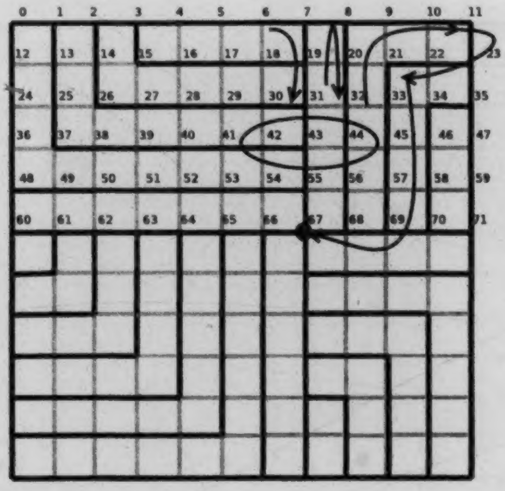

Figure 4.1: Routing around the error blob in HHA

4.2 Simulation Environment

The general simulation configuration has been summarized in Table 4.1. We use our custom designed simulator written in C programming language. Our simulator uses discrete-event simulation models with next-event-time-advance approach [28]. Like our static evaluation in the previous chapter, we use a 12x12 grid (144 nodes) with unit distance $r = 20$m. We place the sink at the center of the grid since this reduces average path length from the sources to the sink. We use 1 Mbps full-duplex bidirectional links. That is, communication can take place in both directions between two neighboring nodes simultaneously at a speed of 1Mbps. We run each simulation for 20 minutes of simulation time and we run 1200 trials of such simulations and take their average. All plots in Section 4.5 show 95% confidence interval for each data point.

Grid Dimension	12 x 12
Grid Unit Distance	20m
Sink Location	Center
Link Bandwidth	1 Mbps
Link Type	Full-duplex
Packet Payload Size	1 Kb
Number of Error Blobs	3
Speed of Error Blobs	15 cm/sec
Blob Axis *a*	20m
Blob Axis *b*	4m
Simulation Time	20 minute
Simulation Trials	1200

Table 4.1: Simulation Configurations

4.2.1 Traffic Model

We use a simple traffic model where each node generates ten packets a second, each with 1Kb payloads. However, in our experiments, we have only one node generating packets, all other nodes work just as forwarding nodes. We do this in order to keep our experiments and analysis simple since our main goal here is to evaluate how resilient our proposed topologies are under different protocols. Note that even if we considered all nodes as traffic generators, we would not have collisions since all communication is on dedicated point-to-point links. However, we would have do deal with congestion on certain parts of the network and lack of buffer space and queuing delay resulting from them. We discuss the issues with multiple sources in the network in Section 4.6 and leave the quantitative evaluation of these issues as future work.

4.2.2 Error Blobs

Optical communication is hampered by obstructions that blocks line of sight. In underwater environments, these obstructions could be underwater organisms, floating objects and sediments. We model these obstructions using ellipses with small minor axis and large major axis, as shown in Figure 3.13 in the previous chapter, and call them *error blobs*. We overlay a separate grid with higher granularity on top of the original network grid to define the movement of the error blobs. This new grid has unit distance of 1 decimeter (0.1 meter) compared to 20m in the original grid. At the start of simulation, we place three error blobs of identical dimension (a x b) on three uniformly chosen random grid points on the new grid with three uniformly chosen random orientations. Each blob moves with a speed of 15cm/sec (unless we experiment with blob speed) within the network in a fashion that is similar to random walk [9]. At each step, the blob moves 1 decimeter in a direction chosen at random from four possible directions (left, right, up, down) unless the blob is currently on the network boundary in which case it choses a direction at random from the two or three possible directions, whichever applicable. The frequency of such steps are chosen so that the overall speed of the blob is 15cm/sec. Note that once we randomly select the orientation of each blob at the beginning, we do not change it again during the simulation, we only change the locations of the blobs. We keep the semi-minor axis of each blob fixed at b = 4m and semi-major axis of each blob fixed at a = 20m. In our experiments with blob size, we keep b fixed at 4m and vary a. At a certain point in time, if a link is inside or intersects one or more of the error blobs, the link is considered down at that moment since the line of sight is lost because of the blob.

4.3 Simulation Metrics

In our simulation, we measure the following metrics for different protocols applied on different topologies under different settings: delivery ratio, average delay per packet and average number of payloads transmitted per successful packet. Each metric is averaged over 1200 trials of simulation.

4.3.1 Delivery Ratio

Delivery ratio is defined by the following equation.

Delivery ratio = (total number of packets successfully delivered at sink/total number of packets sent by the source)

Delivery ratio is the most important metric since it denotes the degree of resiliency supported by a protocol working on a certain topology. It reflects the degree of redundancy inherent in a topology and the degree with which a routing protocol utilizes this redundancy to provide high robustness in the presence of obstacles inside the network.

4.3.2 Average Delay Per Packet (ADPP)

Average Delay Per Packet, abbreviated as ADPP, is defined by the following equation.

ADPP = (total delay faced by all successfully delivered packets/number of packets successfully delivered at sink)

The unit of this metric is seconds/packet. In the above equation, delay of a packet indicates the total time between the generation of the packet and the successful delivery of the packet at the sink. This includes the time spent in transmission and the time spent waiting in the buffer, if at all. We sum up the delays of all successfully delivered packets to get the numerator of the above equation. ADPP indicates the expected time needed to deliver a packet since its generation. It is proportional to the length of the path on which the packet is delivered in the absence of queuing delay.

4.3.3 Average Number of Payloads Transmitted per Successful Packet (APTS)

Average number of Payloads Transmitted per Successful packet, abbreviated as APTS, is defined by the following equation.

APTS = (total number of bits transmitted for all successfully delivered packets)/(number of packets successfully delivered at sink*size of one payload in bits)

The unit of this metrics is payloads/packet. In the above equation, the term "data transmitted" means actual data communication. Thus, if the same packet is transmitted twice at the same forwarding node or at two different hops on its the way to the sink, we sum up both transmissions in our estimate of "total data transmitted". Also, data transmitted for a packet includes not only the transmissions of its payload but also packet overheads and acknowledgments associated with this packet.

Size of one payload is always 1 Kb, *i.e.* 1000 bits. For FLD and DPP protocol, we have fixed length packets whereas for HHA we have variable length packets (since HHA carries path information in the packet). However, the payload field of a packet in all three protocols is 1 Kb in length. See Appendix A.4 for detailed packet formats.

The APTS metric gives an estimate of the amount of communication in a protocol applied on a certain topology and is particularly important for sensor networks since communication is the principal source of energy consumption in a battery-powered node [1]. Note that the APTS metric reflects the amount of communication for successfully delivered packets only. We have also experimented with average number of payloads transmitted per packet in general (considering both delivered and undelivered packets) but do not present our results with such metric since we have observed similar qualitative behavior with this metric as with the APTS metric.

4.4 Experiment Methodology

In this section, we present our assumptions, parameter settings and design of experiments. Implementation details of the three routing protocols have been presented in Appexdix A and verification and validation of the simulation models have been presented in Appendix B. The pseudo-code of the implementations of FLD and HHA protocols have been presented in Appendix C. We do not present the pseudo-code of DPP since it works like FLD except that instead of forwarding packets blindly on all outgoing links, DPP forwards packets on the specific links suggested by the precomputed paths.

4.4.1 Assumptions

We make several assumptions in our implementation of different protocols. We assume infinite buffer space for each protocol since we do not evaluate the effects of limited buffer space in our analysis. We assume that a receiving node can determine whether a packet got corrupted because an obstructing bubble appeared in the line of sight of communication in the middle of a packet transfer. We also assume that all nodes have the complete initial topology stored. Finally, we assume that computation takes place arbitrarily fast. That is, we do not record the time needed for computation (*e.g.*, forwarding logic, flooding table lookup, shortest path computation etc.) in our simulation. For HHA, we also assume that a node can automatically detect the revival of a link once the error blob moves away.

4.4.2 Parameter Settings

The general parameters and their settings are given in Table 4.1. Flooding (FLD) and Dual Paths Protocol (DPP) do not have any special parameter. For Hop-by-Hop Acknowledgement with local update (HHA) protocol, we have three parameters: TTL (Time to Live), ACK_INTERVAL and RETRY_INTERVAL. We perform a calibrating experiment to find out a suitable value for TTL to be used in our other experiments (described in the next subsection).

The value of the parameter ACK_INTERVAL indicates when a timeout for acknowledgment occurs for a DATA packet after it has been transmitted and no ACK has been received. Let us consider the case when node i has just finished the transmission of a DATA packet to node j and is waiting for an ACK from node j. The ACK_INTERVAL parameter should be set to a value that allows node i to wait long enough before deciding that the DATA packet or the ACK from j was lost. In other words, the value of ACK_INTERVAL should be at least equal to the worst-case time needed for node i to receive an ACK from j if no loss occurs. This includes the time for node j to finish transmitting a DATA packet to node i (if node j started this transmission before receiving the DATA packet from node i) and the time for node j to finish transmitting the ACK to node i. Since DATA packets are variable in length because of the variable size of the REMAINING_PATH and LINKS_DOWN field, we should consider the time needed to transmit a maximum-size DATA packet. Assuming a maximum length of REMAINING_PATH field to be 80 (nodes) and a maximum length of LINKS_DOWN field to be be 20 links, the size of such a maximum DATA packet is 2.027 Kb which needs 2.027 msec to be transmitted at 1 Mbps speed. The size of each ACK packet is 27 bits which needs 0.027 msec to be transmitted. Finally, we add a small time delay of 5 microseconds to break ties between events to have a value of 2.059 msec (2.027 + 0.027 + 0.005) as our value for ACK_INTERVAL parameter. After finishing the transmission of a DATA packet, a node waits this long before it decides that the packet did not get through. For detailed packet formats, see Appendix A.4.

Another important parameter for HHA protocol is RETRY_INTERVAL which is the time a node waits before trying again when it finds all its outgoing links to be down at a particular moment according to its LINK_STATUS data structure or it cannot find a path from itself to the sink in the current topology. In our simulation, we set this parameter to be the time that an error blob needs to travel 5 meters if it travels in a straight line. With a blob speed of 15cm/sec, this is equal to 33.33 seconds.

4.4.3 Design of Experiments

We use FLD to demonstrate the inherent redundancy of our proposed deployment topologies and expect a high value of delivery ratio from FLD under a given topology with a minimum delay of delivery. However, FLD causes excessive transmissions that is unsuitable for practical applications, especially for sensor networks. We then use DPP, a simplified representative of multipath protocols, on our topologies as an intermediate solution between FLD and HHA. Finally, we use HHA, a dynamic single-path protocol that exhibit delivery ratio close to FLD with smaller amount of communications on average, even smaller than that of DPP, at the cost of slightly longer delays on average.

We run each simulation trial for a simulation time of 20 minutes and take the average of 1200 such trials to generate each data point in our experiments. We also calculate 95% confidence intervals [19] for each data point and show these intervals in each plot presented in Section 4.5.

For each protocol, we perform three sets of experiments and for each set, we present results with our three metrics (see Section 4.3) on five different topologies (TOP2, TOP3, TOP4, TOP5, TOP5 and TOP6). As mentioned in Section 4.2, we use a single packet generating source node in the entire grid. In the first set of experiments, we vary the shortest-path hop distance of the source node from the sink within the topology under consideration. For each simulation trial for a hop distance, we select at random a node at that hop distance and use that node as the packet generating source. During our experiment with hop distance, we keep the dimensions of error blobs fixed at a = 20m and b = 4m, the speed of error blobs fixed at 15cm/sec, and for HHA, keep TTL fixed at 0.5 sec. The purpose of this experiment is to examine the variation in robustness (delivery ratio), delay (ADPP) and number of transmissions (APTS) as the source node is moved away from the sink. A packet from a source node that is farther away from the sink has higher probability of facing an error blob(s) on its way to the sink because of the increased lengths of the available paths.

In the second set of experiments, we vary the size of the error blobs to see the effects of larger (and smaller) obstructions inside the network. We keep the semi-minor axis fixed at $b = 4$m and vary the length of semi-major axis a. During our experiments with the size of error blobs, we keep hop distance fixed at 10 hops. That is, for each simulation trial, we select at random a node at a hop distance of 10 from the sink within the topology under consideration and use that node as the packet generating source. We keep the speed of error blobs fixed at 15cm/sec and for HHA, keep TTL fixed at 0.5 sec.

In the third set of experiments, we vary the speed of error blobs to see the effects of faster (and slower) obstructions inside the network. As with our experiments with the size of error blobs, we keep hop distance fixed at 10 hops. We also keep the dimensions of error blobs fixed at $a = 20$m and $b = 4$m, and for HHA, keep TTL fixed at 0.5 sec.

For HHA protocol, we perform an extra set of calibrating experiments in which we vary the value of the parameter TTL (Time To Live) in order to find out an appropriate value of TTL (0.5 sec) to use in the above three experiments. In these experiments, we keep hop distance fixed at 10 hops, the dimensions of error blobs fixed at $a = 20$m and $b = 4$m, and the speed of the error blobs fixed at 50cm/sec.

4.5 Analysis of Experimental Results

We present our simulation results and their analysis in this section. Each plot in this section shows 95% confidence intervals for each data points. For an overview of TOP2, TOP3, TOP4, TOP5 and TOP6, please refer to Figure 3.4, Figure 3.11(a), Figure 3.11(b), Figure 3.12(a) and Figure 3.12(b), respectively, and Table 3.1 in the previous chapter.

4.5.1 Analysis of Flooding Protocol (FLD)

Experiment with Hop Distance

In this experiment, we vary the hop distance of the source node from the sink within the topology under consideration in order to examine the effect of hop distance on different metrics under the flooding (FLD) protocol. In this experiment, we keep the dimension of each error blob fixed at $a = 20$m and $b = 4$m and the speed of each error blob fixed at 15cm/sec.

Figure 4.2 shows delivery ratio of flooding (FLD) on different topologies against hop distance of the source node from the sink. As can be seen from Figure 4.2, TOP4, TOP5 and TOP6 have enough redundancy to keep the delivery ratio almost constant with distance since FLD makes sure that a packet goes through all possible paths from source to the sink regardless of where the source is. We also see a significant increase in delivery ratio as we move from TOP4 to TOP5 since in TOP4, a sink edge (edge connected with the sink) remains the single point of failure for the entire quadrant it connects to the sink. TOP5 gets rid of this property by connecting Q_1&Q_4 and Q_2&Q_3 together by two more links. However, as we move from TOP5 to TOP6 by adding two more links the improvement in delivery ratio is very small, although consistent and visible.

In TOP2 and TOP3, each node has exactly two paths and exactly one path, respectively, to the sink. Thus, even with FLD, packets go through exactly one or exactly two paths to the sink. With TOP3 which is a shortest path tree rooted at sink, the farther a node is from the sink, the longer the path that it has to the sink, thus the higher the chance that a packet from this node faces an error blob on its way to the sink. Hence, delivery ratio falls steadily with increased hop distance from the sink (see Figure 4.2). With TOP2, which consists of four undirected Hamiltonian cycles, one path from the source is very small (*e.g.*, 4 or 6 hops) while the other is very long (maximum 32 hops). Thus, most packets are delivered through the primary (shorter) path. Among the packets that are not delivered on this primary path, few get delivered on the secondary (longer) path since this path is very long and is thus more error-prone. As a result, as the primary path gets longer (from 4 to 10 hops), the delivery ratio falls since the secondary path cannot deliver as many packets when the primary path fails to deliver. However, because of the support of the secondary path, the delivery ratio with TOP2 remains higher than that with TOP3 and the delivery ratio falls less steeply with TOP2 than with TOP3 (see Figure 4.2).

Note that with very small hop distance from sink (4 hops), TOP2 outperforms TOP4 in terms of delivery ratio (see Figure 4.2). This is again due to the fact that in TOP4, all paths from a source node go through a single link (the link connected to the sink) which remains the single point of failure. In TOP2, a node has only two paths to the sink but the two paths are geographically distant and are connected to the sink using two different links from the sink. As the hop distance increases from 4, TOP4 keeps its delivery ratio constant because of its high redundancy while the delivery ratio of TOP2 keeps falling steadily below that of TOP4.

Figure 4.3 shows Average Delay Per Packet (ADPP) of flooding (FLD) on different topologies against hop distance from the sink. With 1 Mbps bandwidth and 1 Kb payloads, each packet should optimally be delivered in n milliseconds if the source's best path to the sink is n hops in length. With TOP3 which is a shortest path tree, this is the case since only successful packets are considered in the ADPP metric. However, ADPP is slightly higher than n milliseconds for n hops distance (*e.g.*, 4.2 milliseconds for 4 hops in Figure 4.3) because of the extra time required to transmit the packet overheads (SOURCE, DESTINATION, SEQ etc. fields). As expected, ADPP increases linearly with hop distance for TOP3 (see Figure 4.3). Note that the ADPP metric for FLD (and for DPP) is free from queuing delays since FLD (and DPP) forwards a packet blindly as soon as it receives it (it does not keep a packet in a queue when there is currently no path/link to the sink) and with our packet generation rate (10 1-Kb packets per second), a node never has to keep a packet in the queue while the transmission of the previous packet is in progress on the desired outgoing link.

As can be seen from Figure 4.3, TOP2 has the highest ADPP of all topologies and the ADPP of TOP2 increases almost linearly like TOP3. The high average delay in TOP2 results from the fact that although the primary path from a source to sink in TOP2 has the same number of hops as that in other topologies in Figure 4.3, the secondary path (the rest of the Hamiltonian cycle) is much longer (maximum 32 hops) compared to other topologies. Thus, when the packet is not delivered on the primary path, a very long delay is associated with the packet which increases the average delay. Note that the uncertainty (error bars) with TOP2 is highest in Figure 4.3 compared to other topologies and this is the result of the high difference between the time needed to deliver a packet on the primary path and the time needed to deliver a packet on the secondary path.

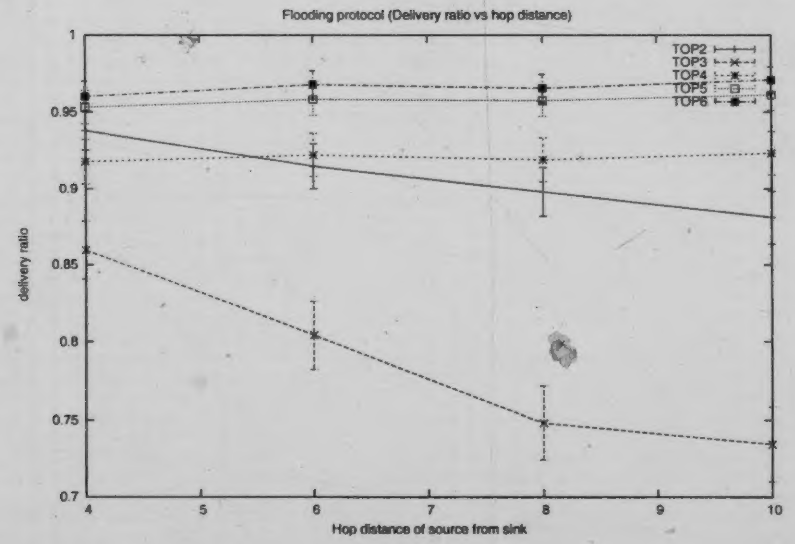

Figure 4.2: Delivery ratio vs hop distance with FLD. a = 20m, b = 4m and blob speed = 15cm/sec

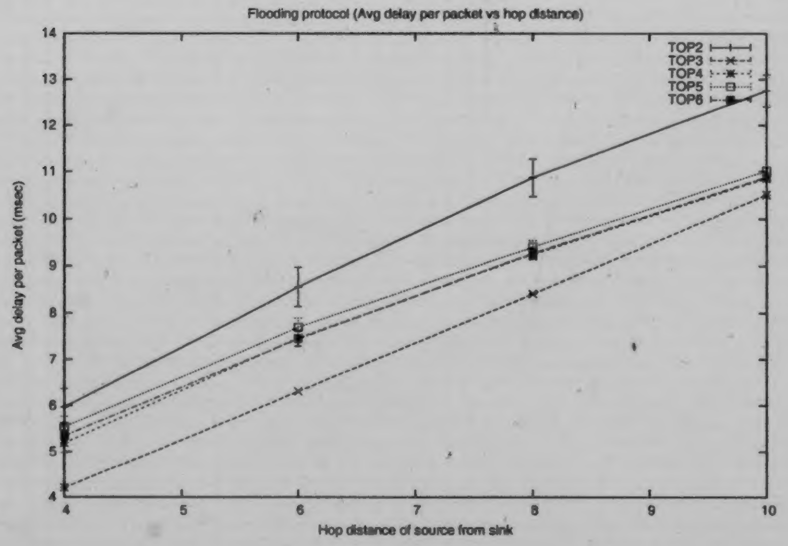

Figure 4.3: ADPP vs hop distance with FLD. a = 20m, b = 4m and blob speed = 15cm/sec

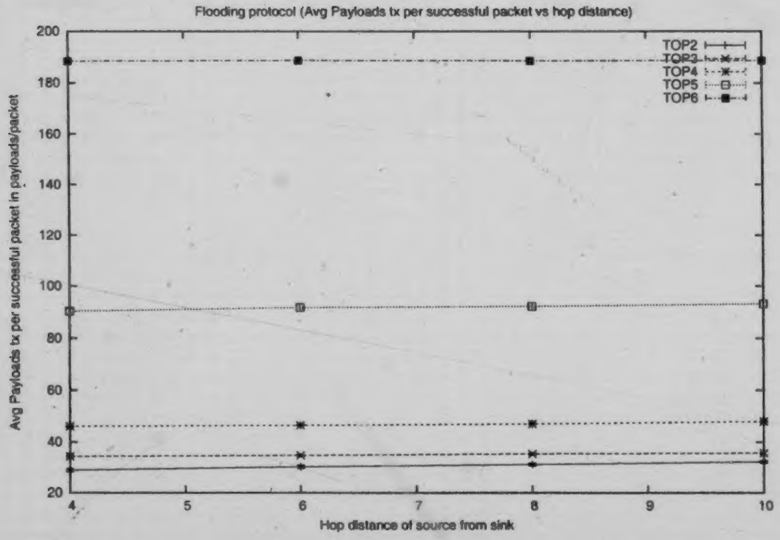

Figure 4.4: APTS vs hop distance with FLD. a = 20m, b = 4m and blob speed = 15cm/sec

As can be seen in Figure 4.3, ADPP of FLD with TOP4, TOP5 and TOP6 stays between that with TOP2 and TOP3. Because of the added redundancies, FLD with TOP4, TOP5 and TOP6 deliver packets on alternative longer paths if they cannot be delivered on the original shortest path and maintain a high level of delivery ratio (see Figure 4.2). This causes an increase in the ADPP compared to that with TOP3. However, alternate paths in TOP4, TOP5 and TOP6 are much shorter than those in TOP2 in most cases, thus these topologies have lower ADPP than that with TOP2. As with TOP2 and TOP3, ADPP for these three topologies increase with hop distance but the slopes decrease with higher hop distances since the ratio of alternate path length to primary path length is smaller for nodes that are far away from the sink in these topologies. In other words, if a packet is not delivered on the primary path, the length of the alternative path compared to the original path is smaller for nodes that are far away from the sink than that for nodes that are closer to the sink (since we add redundant links on the network boundaries).

As can be seen in Figure 4.3, delay with TOP5 is consistently higher that that with TOP4 because TOP5 supports higher delivery ratio by using alternate paths in the adjacent quadrants (since Q_1&Q_4 and Q_2&Q_3 are connected in TOP5) and these paths are longer than the paths that are inside the same quadrant. However, TOP6, which gives the highest delivery ratio (see Figure 4.2), has average delay lower than that with TOP5 and almost the same as that with TOP4. This is due to the fact that since all the quadrants are connected together in TOP6, not only do more paths become available but in many cases better (shorter) alternative paths are found. For example, in TOP5, Q_1 is connected with Q_4 but not with Q_2. In TOP6, Q_1 is connected to both Q_4 and Q_2 (and to Q_3 as well). Thus, for a node in Q_1 for which no path in Q_1 is currently able to deliver a packet and which is very close to Q_2 but is far away from Q_4, FLD with TOP5 will deliver the packet on a very long alternate path that routes through distant Q_4 but FLD with TOP6 will select a relatively very short alternate path through nearby Q_2. Thus, by connecting more adjacent quadrants, TOP6 not only gives higher delivery ratio but also reduces average delay compared with TOP5 and all this is achieved by adding two more links. This justifies the addition of two links on TOP5 to produce TOP6 and demonstrates the superiority of TOP6 over all other topologies.

Figure 4.4 shows Average number of Payloads Transmitted per Successful packet (APTS) of flooding (FLD) on different topologies against hop distance from the sink. For none of the topologies do we see any noticeable change in APTS with varying hop distances from the sink. This is because with FLD a packet is flooded throughout the network regardless of where the source node is located.

Note that for a source that is n hops away form the sink on its shortest path, n payloads need to be transmitted per successful packet in the ideal scenario. However, in Figure 4.4, we see that even for TOP3 which is a shortest path tree, we have almost 35 payloads transmitted for packet on average from a node at a distance of just 4 hops. This is the result of flooding: although a packet is delivered on the only available (shortest) path, the packet is forwarded to the other branches of the tree as well although those branches do not lead to the sink. Moving from TOP3 to TOP4 in Figure 4.4, we see almost 10 more payloads transmitted per successful packet on average. As we move from TOP4 to TOP5 and to TOP6, we see APTS becomes almost double each time. This is because in TOP4, all paths from a source are confined within the quadrant within which the source is located. For TOP5, paths from a source are spread out in two quadrants (since Q_1&Q_4 and Q_2&Q_3 are connected) and for TOP6, paths from a source are spread out in four quadrants (since all quadrants are now connected). As can be seen in Figure 4.4, with FLD on TOP6, for a successfully delivered packet from a source at a distance of just 4 hops, almost 190 payloads are transmitted. This clearly indicates that FLD is not a suitable protocol to use if we want to save energy. However, we demonstrate later that we can use other protocols (HHA) to achieve almost similar delivery ratio as FLD with much fewer transmissions at the expense of slightly increased delay.

As can be seen in Figure 4.4, TOP2 has considerably lower APTS than that of TOP3 although TOP2 has two disjoint paths from each source (thus higher delivery ratio). This is because with flooding in a cycle (TOP2), if a packet faces an error blob at some point, it does not get forwarded any further in that direction to the sink, thus restricting transmissions. With FLD in a tree rooted at sink (TOP3), if a packet faces an error blob at a forwarding node, it does not get forwarded on one or more subtrees of that node but it is forwarded on all other subtrees and branches, causing higher APTS than that in TOP2. Thus, if FLD is to be used, TOP2 is a good candidate in terms of number of transmissions.

To summarize, FLD utilizes all possible paths in the network to provide very high degree of delivery ratio at minimum possible delay but incurs excessive overhead. TOP4, TOP5 and TOP6 have enough redundancy to maintain almost constant delivery ratio at different hop distances. TOP2 and TOP3, on the other hand, have very limited number of paths to the sink and, therefore, their delivery ratios fall down as the source is moved away from the sink. Finally, TOP6 not only provides the best delivery ratio but also introduces better alternative paths in the network and thus have lower average delay than that of TOP5.

Experiment with Error Blob Size

In this experiment, we vary the size of the error blobs to see the effect of larger and smaller obstructions in the network under the flooding (FLD) protocol. We keep the semi-minor axis fixed at $b = 4$m and vary the length of semi-major axis a. In this experiment, we keep the hop distance of the source node from the sink fixed at 10 hops and the speed of each error blob fixed at 15 cm/sec.

Figure 4.5 shows delivery ratio of flooding (FLD) on different topologies against error blob size. As expected, delivery ratio of FLD with each topology decreases with larger error blobs. The rate of this fall is highest and almost linear with TOP3 and TOP2 which have just one and just two paths to the sink from each source, respectively. As we move from TOP4 to TOP5 by connecting $Q_1\&Q_4$ and $Q_2\&Q_3$ with two more links, delivery ratio increases significantly and the difference gradually gets larger as the size of the error blobs grow. With TOP6, the decrease of delivery ratio with larger blobs is the minimum of all other topologies and even with a = 30m, the delivery ratio is 95% which is significantly higher than that of TOP5 at this blob size. This suggests that with larger error blobs, TOP6 is preferable over TOP5 (and other topologies) in terms of delivery ratio since the speed of the error blobs does not affect the delivery ratio on average (shown later).

Figure 4.6 shows Average Delay Per Packet (ADPP) of flooding (FLD) on different topologies against error blob size. With TOP3 (shortest path tree), the delay is minimum since packets are delivered on the shortest path only. Delay with TOP3 remains constant with varying blob size since we consider only successful packets in the ADPP metric and successful packets are always delivered on the shortest path in TOP3. ADPP with TOP2 is the highest while ADPP with TOP4, TOP5 and TOP6 remains in between the ADPPs of TOP2 and TOP3. As with our experiment with hop distance (see Figure 4.3), TOP6 shows better ADPP than TOP5 for different blob sizes because of the availability of better paths through adjacent quadrants, although TOP6 provides better delivery ratio.

As can be seen in Figure 4.6, ADPP with TOP2, TOP4, TOP5 and TOP6 increases with increasing blob sizes since larger blobs increase the probability that the primary path fails and the packet gets delivered on alternate longer paths. However, the rate of increase of ADPP falls down with larger error blobs for TOP2 and TOP4 but goes up with larger blobs for TOP5 and TOP6. With TOP2 and TOP4, the number of possible paths from a source to the sink is limited and they are all confined within the same quadrant. For TOP2, we have only two paths. Thus, with increasing blob sizes, although more and more packets are dropped undelivered, the successful packets are delivered on the limited number paths within the same quadrant. Therefore, the increase of delays are less acute at higher blob sizes. With TOP5 and TOP6, quadrants are connected together and thus packets can be delivered on paths that go through adjacent quadrant (for TOP5) or quadrants (for TOP6) if they cannot be delivered on the path within the same quadrant as the one in which the source is located. With larger error blobs, TOP5 and TOP6 use these increasingly longer paths, thus they maintain high delivery ratio but the average delay keeps increasing with higher rates as the size of the error blobs increases. With a = 10m, TOP4 and TOP6 have approximately the same ADPP. However, because of the different increase patterns, TOP6's ADPP reaches 11.25 msec at a = 30m while TOP4's ADPP reaches slightly less than 11 msec at a = 30m. However, as long as error blobs are small enough with $a \leq$ 20m, TOP6 remains the best choice in terms of delivery ratio and delay.

Figure 4.7 shows Average number of Payloads Transmitted per Successful packet (APTS) of flooding (FLD) on different topologies against error blob size. Relative performances of the five topologies remain the same as in experiment with hop distance (see Figure 4.4). However, unlike our experiments with hop distance, APTS of FLD on different topologies decrease very slowly with increased blob sizes. Larger blobs occasionally prevent some part of the network from being flooded which results in this behavior.

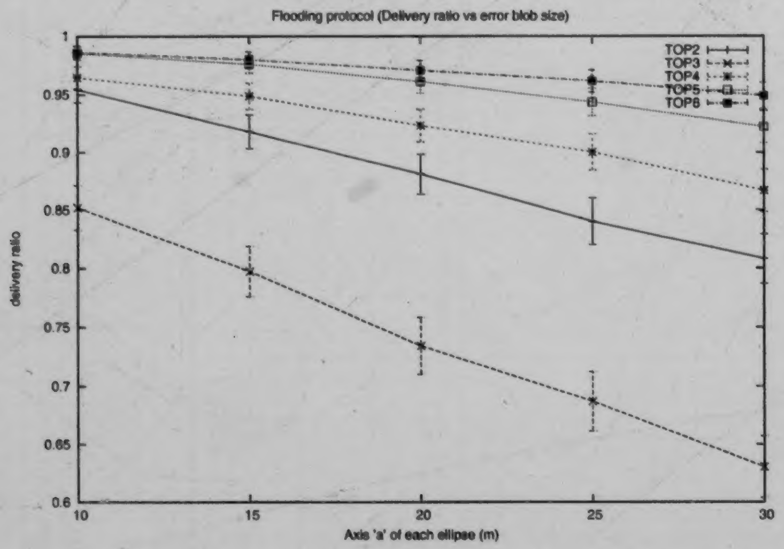

Figure 4.5: Delivery ratio vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec

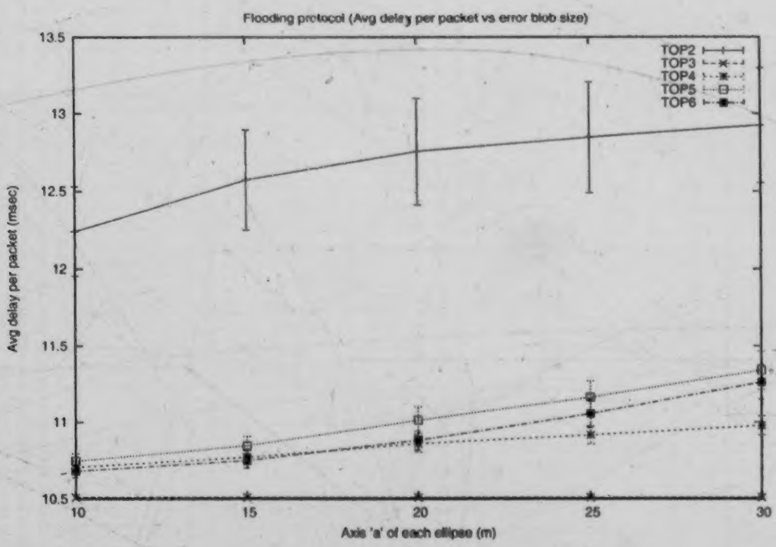

Figure 4.6: ADPP vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec

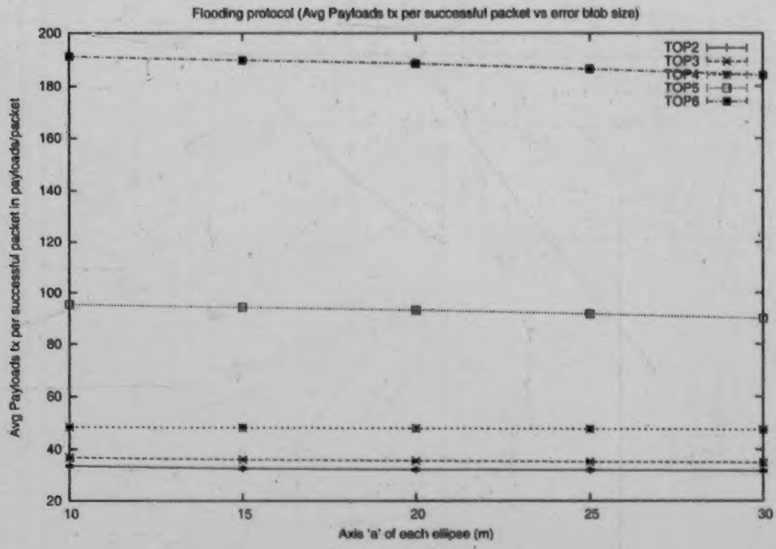

Figure 4.7: APTS vs error blob size with FLD. Hop distance = 10 and blob speed = 15cm/sec

To summarize, larger error blobs cause delivery ratio of each topology to decrease with FLD because of the increased degree of obstruction inside the network. However, TOP6 retains almost 95% delivery ratio with FLD even with error blobs of dimension as large as a = 30m and b = 4m. While maintaining this high delivery ratio, TOP6 introduces longer delays at larger blob sizes but the average delay with TOP6 remains lower than that with TOP5 because of the availability of better paths in TOP6 through adjacent quadrants.

Experiment with Error Blob Speed

In this experiment, we vary the speed of the error blobs to see the effect of faster and slower obstructions in the network under the flooding (FLD) protocol. Here, we keep the hop distance of the source node from the sink fixed at 10 hops and the dimension of each error blob fixed at a = 20m and b = 4m.

Figure 4.8, Figure 4.9 and Figure 4.10 show delivery ratio, ADPP and APTS, respectively, of FLD on different topologies against the speed of error blobs. As can be seen from these figures, none of these metrics shows any noticeable variation with the change of speed of error blobs. With the size of error the blobs constant at a = 20m and b = 4m, the total area occupied by these blobs in the network remains constant even if they move at a different speed. Therefore, the values of different metrics remain approximately the same on average with changing speeds of the error blobs. Similar behavior has been observed in our experiments with blob speed with other protocols (DPP and HHA). Therefore, we do not present the numeric results of our experiments with blob speed with DPP and HHA protocols.

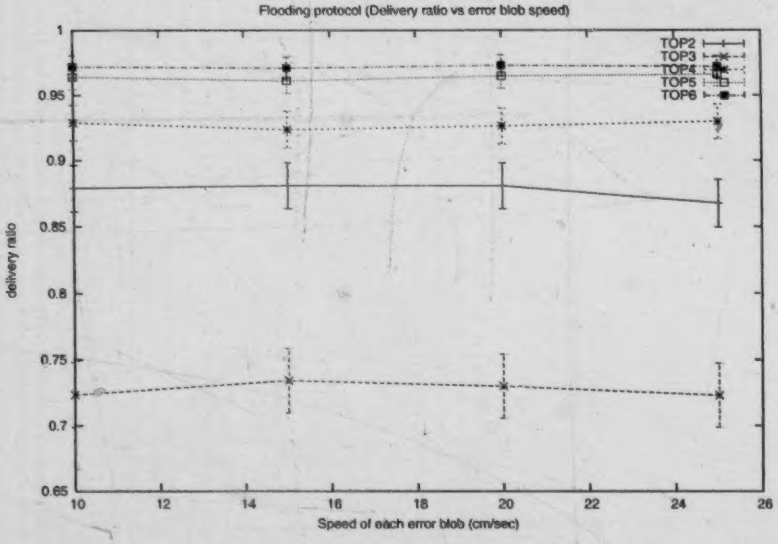

Figure 4.8: Delivery ratio vs error blob speed with FLD. a = 20m, b = 4m and Hop distance = 10

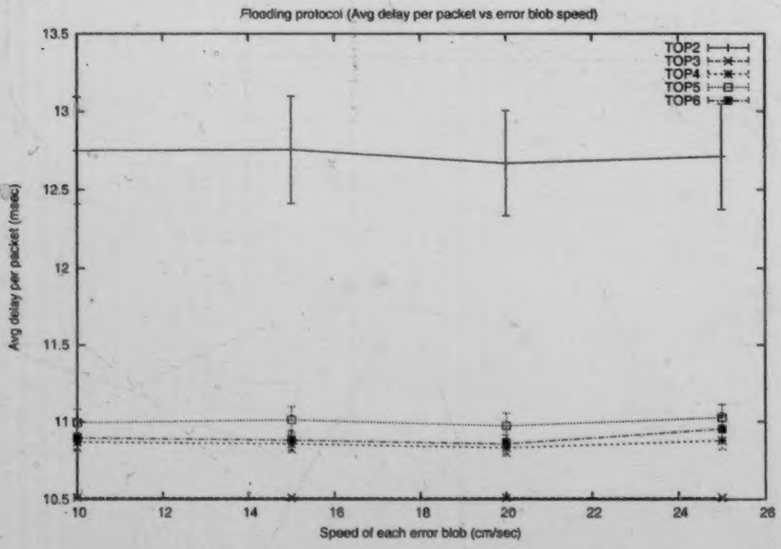

Figure 4.9: ADPP vs error blob speed with FLD. a = 20m, b = 4m and Hop distance = 10

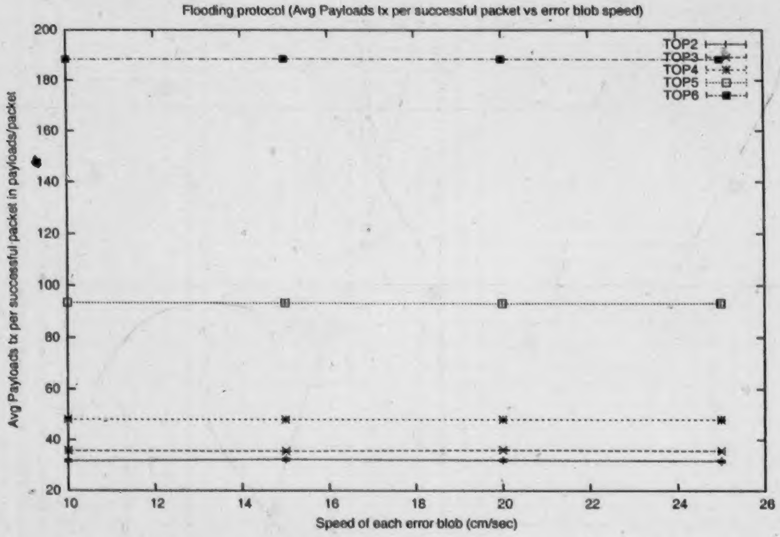

Figure 4.10: APTS vs error blob speed with FLD. a = 20m, b = 4m and Hop distance = 10

4.5.2 Analysis of Dual Paths Protocol (DPP)

Experiment with Hop Distance

In this experiment, we vary the hop distance of the source node from the sink within the topology under consideration in order to examine the effect of hop distance on different metrics under the Dual Paths Protocol (DPP). In this experiment, we keep the dimension of each error blob fixed at a = 20m and b = 4m and the speed of each error blob fixed at 15cm/sec.

Figure 4.11 shows delivery ratio of Dual Paths Protocol (DPP) on different topologies against hop distance from the sink. As can be seen from Figure 4.11, TOP2 and TOP3 show exactly the same delivery ratio in DPP protocol as we saw in FLD (see Figure 4.2) because these two protocols use the same path(s) to deliver a packet. TOP4 consistently shows smaller delivery ratio compared to TOP2, TOP5 and TOP6. The latter three topologies show similar performance since they use two completely disjoint paths from the source to the sink, whereas with TOP4 the two selected paths from the source go through the same link at the last hop to the sink. Unlike with FLD where TOP4, TOP5 and TOP6 keep delivery ratio constant with varying hop distance by fully utilizing the inherent redundancy of the topologies, with DPP each of these topologies faces decrease in delivery ratio as the source is moved away from the sink since longer paths increase the possibility of facing error blobs. However, as we move from TOP2 to TOP5 and from TOP5 to TOP6, the rate of this decrease falls down and TOP6 shows visibly better performance than TOP5 (and TOP5 better than TOP2) as the hop distance increases. This is because with TOP2 the alternative path is extremely long compared to the primary path and thus more error-prone. Thus, the alternative path cannot offset and keep up with all the losses of the primary path and the nature of delivery ratio mostly indicates the performance of the primary path which falls with increased distance. With TOP5 and TOP6, the alternative paths are increasingly better (shorter) than those in TOP2 and these paths have lengths closer to the lengths of the primary paths. Thus, the alternative paths in TOP5 and TOP6 can better offset and keep up with the losses of the primary paths and the delivery ratio falls less steeply with increasing hop distance. This observation suggests that in addition to selecting alternative paths that are disjoint and geographically distant, we should also make sure that the alternative paths have lengths comparable to the primary path. Since we have significantly higher number of nodes at higher hop distances than at lower hop distances from the sink, TOP6 is the most promising topology again in terms of delivery ratio using DPP.

Figure 4.12 shows Average Delay Per Packet (ADPP) of Dual Paths Protocol (DPP) on different topologies against hop distance from the sink. Like delivery ratio, average delay of TOP2 and TOP3 remains the same as with FLD (see Figure 4.3) since DPP and FLD have the same paths to deliver a packet under these topologies. With TOP4, TOP5 and TOP6, average delay of DPP (Figure 4.12) is slightly lower than that of FLD (Figure 4.3), especially at lower hop distances from the sink. This is due to the higher delivery ratio in FLD which causes increasingly longer paths for delivery and the alternate paths become even longer compared to the primary path as the source is moved closer to the sink (since we add redundant links on the network boundaries). However, as can be seen from Figure 4.3 and Figure 4.12, the difference in average delay is so small that it becomes justified to use the redundant paths to improve delivery ratio. FLD does that but it causes an enormous amount of transmissions in the network. We shall recover from this excessive transmissions property in our next protocol (HHA).

Figure 4.13 shows Average number of Payloads Transmitted per Successful packet (APTS) of Dual Paths Protocol (DPP) on different topologies against hop distance from the sink. With DPP, packets are not flooded anymore, thus we see a huge reduction in APTS in Figure 4.13 compared to the APTS of FLD in Figure 4.4. TOP3 has the lowest APTS since packets are transmitted along a single path which is the shortest path. Note that APTS of DPP with TOP3 is much lower than the APTS with TOP3 we saw in FLD (Figure 4.4) since although FLD delivered packets on the same path, it unnecessarily forwarded packets on the other branches of the tree. As can be seen in Figure 4.13, TOP2 has the highest APTS since the alternate path is extremely long. APTS with TOP4, TOP5 and TOP6 falls from approximately 50, 90 and 190 payloads per packet, respectively, in FLD (Figure 4.4) to approximately 20 payloads per packet in DPP (Figure 4.13). Since the path lengths get longer with increased hop distance, we see a steady increase in APTS in Figure 4.13 with higher hop distance for all five topologies. As we move from TOP4 to TOP5 and from TOP5 to TOP6, we see consistent reduction in APTS since the length of the alternate path becomes shorter (while the length of the primary path is the same for all three topologies). The differences between these three topologies decreases with increasing hop distance since the length of the alternative path as compared with the length of the primary path reduces with higher hop distances from the sink. Overall, unlike FLD, APTS is now a function of the path lengths of the primary and the alternative paths instead of the network size.

To summarize, Dual Paths Protocol (DPP) fails to utilize the redundancies inherent in TOP4, TOP5 and TOP6 because of its static nature. Therefore, delivery ratio of DPP on these topologies decrease as the source is moved away from the sink. However, TOP6 still shows the best delivery ratio at hop distances higher than 6 since it supports shorter alternate paths. We do not see any significant change in average delay with DPP compared to FLD since DPP delivers packets only on either of the two shortest paths from the source to the sink. However, we see a huge reduction in average number of payloads transmitted per packet in DPP compared to FLD since DPP does not flood a packet throughout the network. With DPP, number of transmissions is proportional to the lengths of the primary and the alternate path instead of the size of the network.

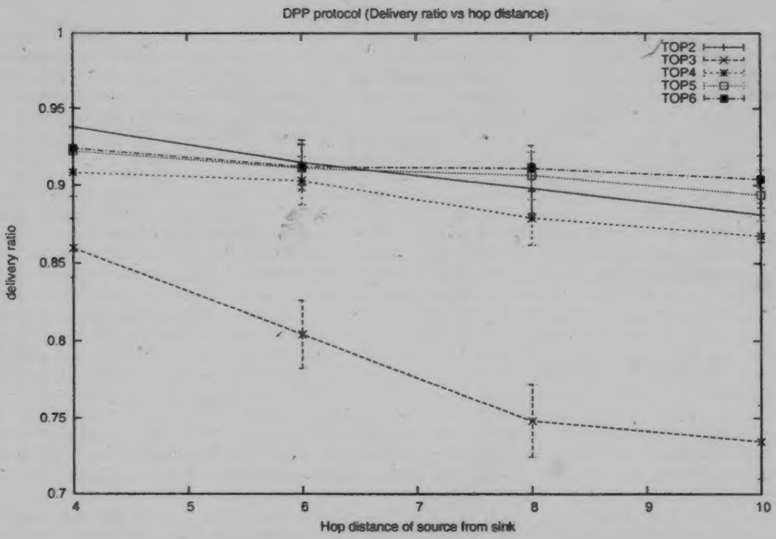

Figure 4.11: Delivery ratio vs hop distance with DPP. a = 20m, b = 4m and blob speed = 15cm/sec

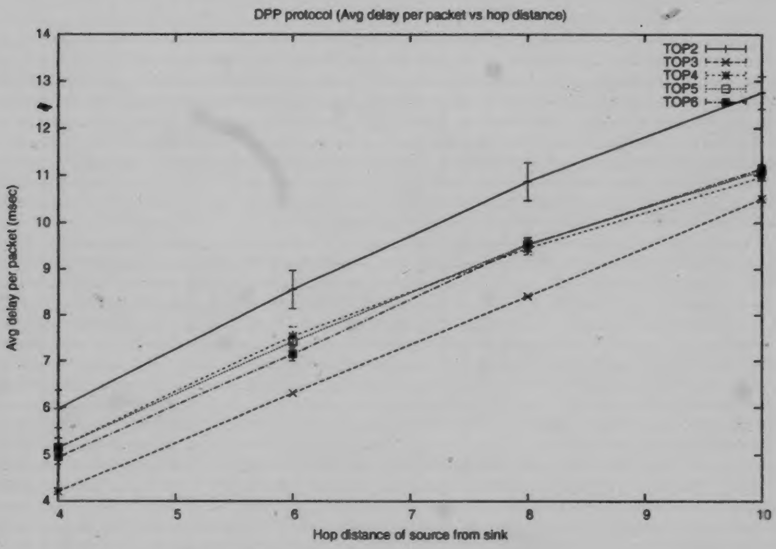

Figure 4.12: ADPP vs hop distance with DPP. a = 20m, b = 4m and blob speed = 15cm/sec

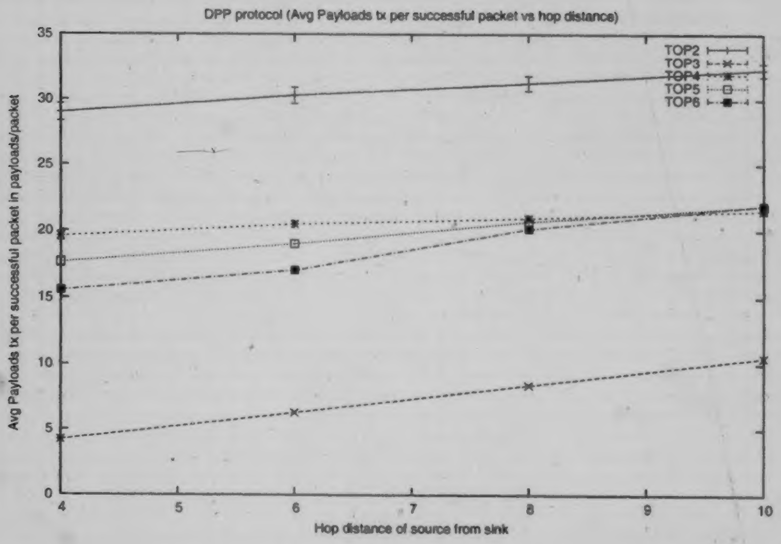

Figure 4.13: APTS vs hop distance with DPP. $a = 20$m, $b = 4$m and blob speed = 15cm/sec

Experiment with Error Blob Size

In this experiment, we vary the size of the error blobs to see the effect of larger and smaller obstructions in the network under the Dual Paths Protocol (DPP). We keep the semi-minor axis fixed at $b = 4$m and vary the length of semi-major axis a. In this experiment, we keep the hop distance of the source node from the sink fixed at 10 hops and the speed of each error blob fixed at 15 cm/sec.

Figure 4.14 shows delivery ratio of DPP on different topologies against error blob size. As can be seen from Figure 4.14, delivery ratio of TOP2 and TOP3 with DPP remains exactly the same as with FLD (Figure 4.5) for different blob sizes. However, unlike FLD, delivery ratio of TOP4, TOP5 and TOP6 decreases almost as steeply as TOP2 with increased blob sizes because DPP uses only two paths when applied on these topologies. However, TOP5 and TOP6 consistently maintains better delivery ratio because these topologies use shorter, thus less error-prone, paths to the sink. The graph in Figure 4.14 shows how DPP fails to utilize the inherent redundancy of TOP4, TOP5 and TOP6 by statically selecting two fixed paths from each source to the sink. A more intelligent multi-path routing protocol would dynamically select the two paths to route a packet to consider current network conditions and would prevent the delivery ratio from falling sharply with increased blob sizes by utilizing the redundancy. However, the delay (ADPP) and transmissions (APTS) incurred by the DPP protocol (discussed next) show us representative values that we can expect from an intelligent disjoint multi-path protocol [30, 50] since DPP always uses two "shortest" disjoint paths (whenever possible) from each source. We shall see later that a single path reliable protocol (HHA) can achieve high delivery ratio with fewer transmissions and acceptable increase in delay on average.

Figure 4.15 shows Average Delay Per Packet (ADPP) of DPP on different topologies against error blob size. Like delivery ratio, average delay of TOP2 and TOP3 with DPP remains exactly the same as with FLD (Figure 4.6) for different blob sizes. As can be seen from Figure 4.15, ADPP of TOP4, TOP5 and TOP6 remains between that of TOP2 and TOP3 and increases almost at the same rate as that of TOP2 with increased blob sizes since with each of these topologies, increased blob sizes mean increased failure probability along the primary path and thus increased rate of delivery on the alternate path. Note that with FLD, ADPP of TOP5 and TOP6 increased at much higher rate with increased blob sizes (see Figure 4.6) since a very high delivery ratio was maintained by delivering the packet on more and more longer paths in presence of failure on primary (thus, shortest) paths. As with FLD, TOP6 shows lower ADPP than that of TOP5 because the selected alternate paths are shorter in TOP6 in some cases.

Figure 4.16 shows Average number of Payloads Transmitted per Successful packet (APTS) of DPP on different topologies against error blob size. As with our experiment with hop distance (see Figure 4.13), we see a huge reduction in APTS in Figure 4.16 compared to the APTS of FLD (Figure 4.7) for TOP3, TOP4, TOP5 and TOP6 since packets are not flooded anymore. Also, APTS of TOP4, TOP5 and TOP6 remains between that of TOP2 and TOP3. However, we see a gradual decrease of APTS with increased blob sizes for TOP2, TOP4, TOP5 and TOP6 since with increased blob sizes, although the packet is delivered on one of the two selected paths, the probability that the packet gets lost and not forwarded further on the other path increases which reduces APTS. APTS remains constant with blob sizes for TOP3 since packets are delivered on a single path and we consider only successful packets in this metric.

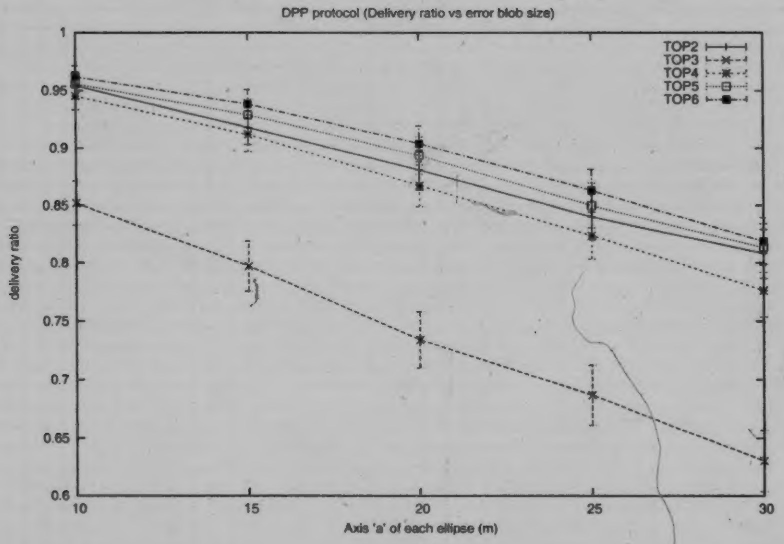

Figure 4.14: Delivery ratio vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec

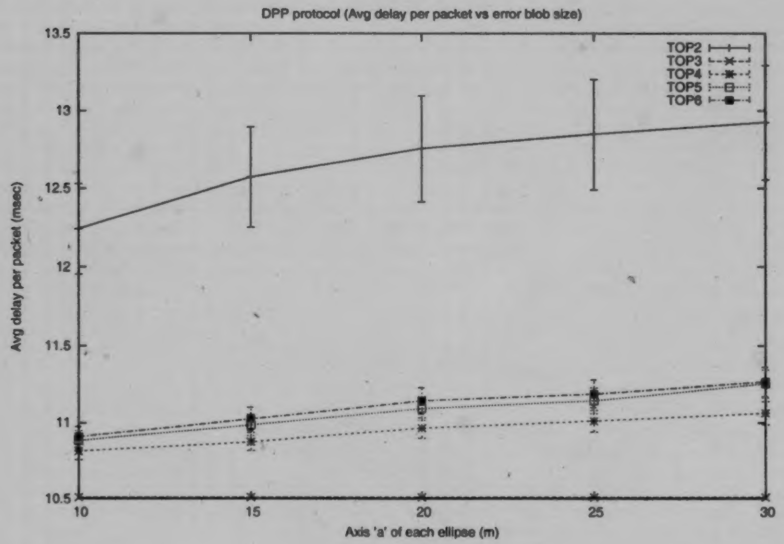

Figure 4.15: ADPP vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec

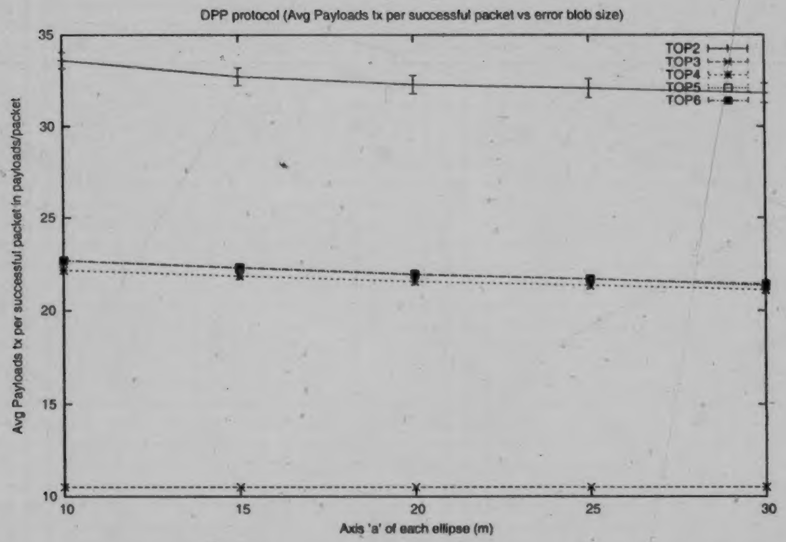

Figure 4.16: APTS vs error blob size with DPP. Hop distance = 10 and blob speed = 15cm/sec

To summarize, larger error blobs cause higher reduction in delivery ratio with DPP than with FLD on TOP4, TOP5 and TOP6 since DPP fails to utilize the redundancy in these topologies. A more intelligent multi-path routing scheme would dynamically select the paths to maintain higher delivery ratio with increased blob sizes. However, average delay (ADPP) and communication overhead (APTS) metrics with DPP remains representative of the similar metrics with an intelligent disjoint multi-path scheme [30, 50] since DPP uses two disjoint "shortest" paths for packet delivery.

4.5.3 Analysis of Hop-by-Hop Acknowledgment Protocol (HHA)

Experiment with TTL

In this experiment, we vary the Time To Live (TTL) parameter of the Hop-by-Hop Acknowledgment with local update (HHA) protocol to find out a suitable value of this parameter to be used in the subsequent experiments where we vary hop distance and error blob size. The value of TTL should be large enough so that there is enough time for HHA to deliver packets on recomputed/rerouted paths, if any, in case the packet cannot be delivered on the original path because of one or more error blobs. Having larger values of TTL will further improve delivery ratio by keeping the packets longer in the queue when there is no available path to the sink and route the packets when the error blob(s) moves away to make one or more paths available again. However, this would also introduce longer average delays. In this experiment, we use different values of TTL to make the resiliency/delay trade-off and find out a value of TTL for HHA that balances these two factors. We keep hop distance fixed at 10 hops, the dimensions of error blobs fixed at $a = 20$m and $b = 4$m, and the speed of the error blobs fixed at 50cm/sec.

Figure 4.17 shows delivery ratio of HHA on different topologies against TTL (time to live). As can be seen, delivery ratio with each topology gradually increases as we increase TTL. This is because with longer TTL, when a packet gets stuck at some node (because no path to the sink currently exists from the node), the packet waits in the PACKET_BUFFER for a longer time before it is dropped, thus the probability that the blocking error blob(s) moves enough to make one or more paths available again so that the packet can be forwarded increases. This improves the delivery ratio on average. This improvement in delivery ratio with TTL becomes more prominent as we move from the topologies with higher degrees of redundancy to the the topologies with lower degrees of redundancy (*e.g.*, from TOP5 to TOP4 and from TOP2 to TOP3) since packets have higher probability of getting stuck in topologies with lower degrees of redundancy and thus higher values of TTL help these topologies more in achieving higher delivery ratio. As can be seen in Figure 4.17, the improvement in delivery ratio with higher TTL values is very slow. With TOP3, delivery ratio increases from 74% to only 78% even when we increase TTL from only 0.5 sec to as high as approximately 400 sec. This is the result of the slow speed of the physical objects and their random walk movement pattern. A speed of 50 cm/sec, which is a very high speed for a physical object or organism under water, with a random walk pattern is often not enough for a large error blob to move away sufficiently even in 400 sec.

Figure 4.18 shows Average Delay Per Packet (ADPP) of HHA on different topologies against TTL. As can be seen, ADPP increases rapidly with increasingly higher rate as we increase TTL and the increase is higher for topologies that experience higher improvement in delivery ratio with TTL in Figure 4.17 (*e.g.*, TOP3 and TOP2). With TTL increased to a higher value, packets stuck at a node can wait longer in the PACKET_BUFFER before being dropped. If the error blob clears away before the TTL expires and such a packet gets delivered to the sink, the delay associated with the packet is very high since it waited in the PACKET_BUFFER for a very long time (close to TTL). This increases the average delay rapidly. Note that when a packet gets stuck at a node, all the subsequent packets that arrives at this node are placed in the PACKET_BUFFER behind this packet. Once the error blob clears away, packets are forwarded from the front of the buffer. Thus, the total time spent in the PACKET_BUFFER is the sum of the time spent waiting for the error blob to move away and the time spent waiting for the packets ahead in the buffer to be cleared once the error blob is gone. Note in Figure 4.18 that the uncertainty (error bars) of ADPP goes up with higher values of TTL. This is due to the huge difference between the delays of the packets that get delivered without having to wait in a PACKET_BUFFER and the packets that get delivered after waiting a while in a PACKET_BUFFER for the error blob(s) to move away. For the former class of packets, delay is the time required to transmit the packet along the selected path which is a function of the length of the delivery path and the bandwidth of the links. On the other hand, for the latter class of packets, delay consists mostly of the time required for the "physical error blob" to move away and the speed of physical objects is much lower than the speed of communication. Overall, increasing TTL gives very small increase in delivery ratio at the cost of very high average delay per packet. Therefore, having very high values for TTL is not justified, especially for applications for which delay needs to be kept below a certain level (*e.g.*, real-time applications).

Figure 4.19 shows Average number of Payloads Transmitted per Successful packet (APTS) of HHA for different topologies against TTL. We do not see any noticeable change in APTS with increasing TTL since packets waiting in the buffer do not incur transmissions.

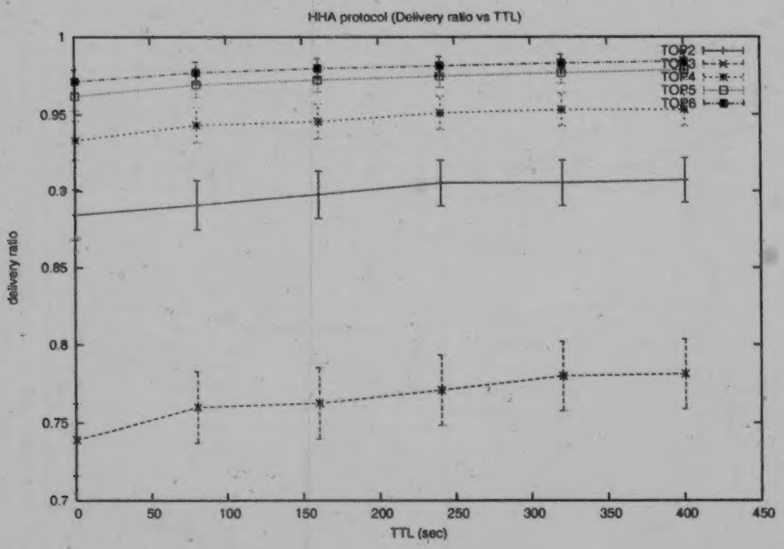

Figure 4.17: Delivery ratio vs TTL with HHA. a = 20m, b = 4m, hop distance = 10 and blob speed = 50cm/sec

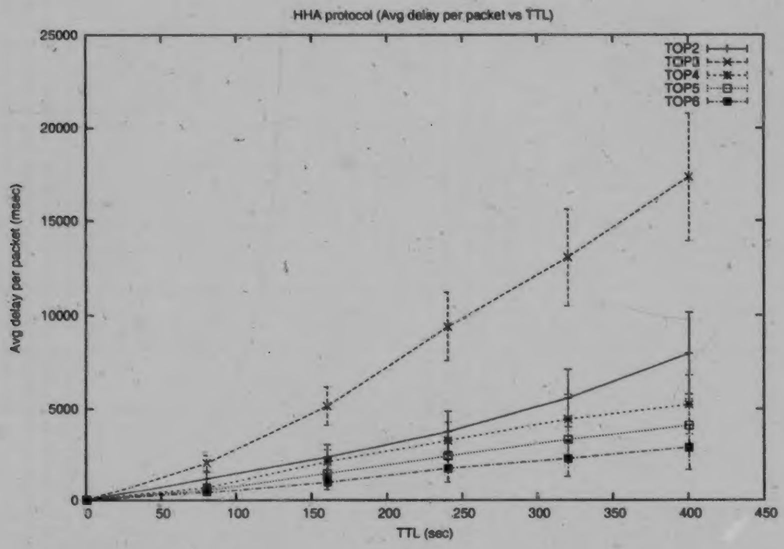

Figure 4.18: ADPP vs TTL with HHA. a = 20m, b = 4m, hop distance = 10 and blob speed = 50cm/sec

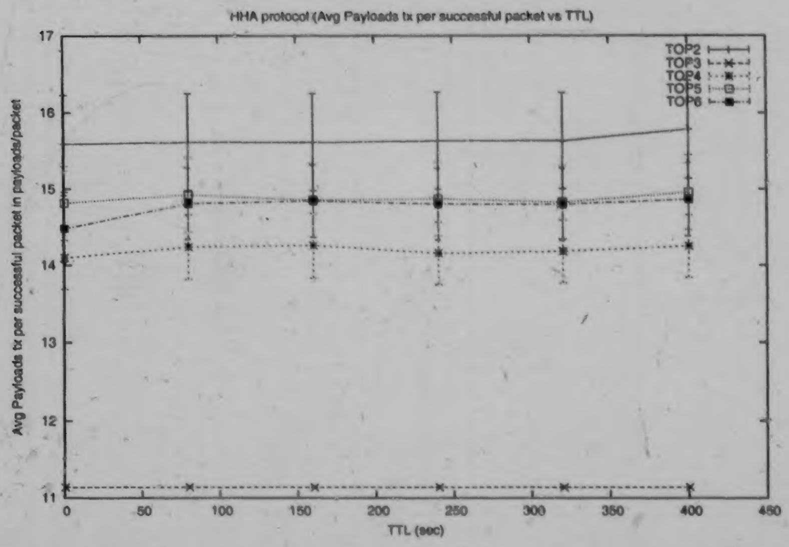

Figure 4.19: APTS vs TTL with HHA. $a = 20$m, $b = 4$m, hop distance = 10 and blob speed = 50cm/sec

In summary, increasing the value of TTL above 0.5 sec provides very slow increase in delivery ratio because it allows packets to wait longer in the queues before being dropped but results in a very high average delay per packet due to the long delays in the queues. Therefore, we use 0.5 sec as a reasonable value for TTL in our next experiments. Note that we have kept the speed of error blobs fixed at 50 cm/sec in this experiment which is higher than the speed we use in our subsequent experiments (15 cm/sec). We do this in order to make the effect of TTL more conspicuous. We have also experimented with an error blob speed of 15 cm/sec (not shown in this thesis) and found out that the increase in delivery ratio with higher TTL is even slower with this speed since it takes longer in such settings for an error blob to move away sufficiently. This justifies the choice of a small value for TTL (0.5 sec).

Experiment with Hop Distance

In this experiment, we vary the hop distance of the source node from the sink within the topology under consideration in order to examine the effect of hop distance on different metrics under the Hop-by-Hop Acknowledgment with local update (HHA) protocol. In this experiment, we keep the dimension of each error blob fixed at $a = 20$m and $b = 4$m, the speed of each error blob fixed at 15cm/sec and TTL fixed at 0.5 sec.

Figure 4.20 shows delivery ratio of Hop by Hop Acknowledgment with local update protocol (HHA) on different topologies against hop distance from the sink. As can be seen from Figure 4.20, delivery ratio of HHA very closely mimics that of FLD (Figure 4.2) and there is no significant difference between the two protocols in terms of this metric for all five topologies. This is because like FLD, HHA takes advantage of the redundancies inherent in the topologies, although in a different way. HHA does this by selecting an alternate "remaining path" whenever it sees that a packet cannot be forwarded any further on the original path and this process repeats at each intermediate node until the packet reaches the sink or TTL is expired. Note that if TTL is infinite, the delivery ratio for all topologies should approach 100% since packets are not dropped, rather they are kept in the PACKET_BUFFER until a path to the sink becomes available again (because the blocking error blob has moved away). However, we have seen in our experiment with TTL that this increase in delivery ratio is very slow with higher values of TTL and incurs excessive delay. Therefore, in the graph shown in Figure 4.20, we keep TTL fixed at 0.5 sec which in most cases is enough time for a packet to be rerouted around an error blob but is shorter than the time required for an error blob that is blocking all possible paths to the sink for a packet to move enough so that one or more paths to the sink become available again. As a result, in most cases, delivered packets in this HHA settings are those packets that were not stuck at a node for some time on their way to the sink because all possible paths from that node were blocked. Therefore, the delivery ratio is approximately the same as that in FLD. Note that although HHA exhibits similar delivery ratio to FLD, it achieves this delivery ratio in a different way, *i.e.*, using a single but often longer dynamically adjusted path. Thus, delay and transmissions characteristics of HHA is expected to be different which we investigate next.

Figure 4.21 shows Average Delay Per Packet (ADPP) of HHA protocol on different topologies against hop distance from the sink. HHA uses a single path for each packet delivery and whenever a link on the path is broken, a new remaining path from the current node is selected. This makes the recomputed paths considerably longer than an alternate path selected from the source of the packet. Thus, HHA has higher average delay (see

Figure 4.21) than that of FLD (Figure 4.3), especially in TOP4, TOP5 and TOP6 which have the maximum redundancy. ADPP with HHA on TOP3 is only slightly higher than that with FLD since the presence of only one path keeps HHA from recomputing long suboptimal alternative paths. This slight increase is thus due to the extra time needed to transmit the packet headers which are longer and variable in HHA. With TOP2, TOP4, TOP5 and TOP6, we see in Figure 4.21 a noticeable increase in ADPP compared to FLD (Figure 4.3) because of locally recomputed paths. However, a small portion of this increased delay is caused by long and variable length packet headers and thus can be avoided by designing a more intelligent protocol that avoids such header formats. Another very small portion of this delay is caused by the time that a forwarding node has to wait for an ACK before it decides that the link is down and performs re-computation. With a TTL of 0.5 sec, packets that are stuck at a node (because there is currently no path to the sink) and wait in the PACKET_BUFFER to be forwarded are usually dropped eventually because their TTL expires before the error blob(s) moves away. Thus, the ADPP shown in Figure 4.21 hardly contains any delay caused by such waiting in the the buffer. As in FLD, TOP6 exhibits better ADPP than that of TOP5 even though TOP6 has higher delivery ratio since TOP6 has better alternate paths in many cases because of the added connectivity between adjacent quadrants. As can be seen in Table 4.2 which summarizes the performance of the three protocols on our best topology TOP6 at a hop distance of 10, TOP6 has an ADPP of 14.24 msec with HHA compared to 10.88 msec and 11.14 msec ~~with~~ FLD and DPP, respectively.

Figure 4.22 shows Average number of Payloads Transmitted per Successful packet (APTS) of HHA protocol on different topologies against hop distance from the sink. Since HHA delivers each packet on a single path, both delay and number of transmissions are proportional to the length of the path on which a packet is delivered. Thus, both ADPP and APTS should exhibit similar behavior. As can be seen from Figure 4.22, APTS for different topologies show similar relative behavior as ADPP (Figure 4.21) for different hop distances. Note that we include the transmissions of ACK in our metric which is negligible compared to the sizes of the packets. As can be seen from the summary in Table 4.2, for our best topology TOP6, APTS at a hop distance of 10 is 14.32 payloads per packet for

HHA compared to 188.60 payloads per packet for FLD and 21.96 payloads per packet for DPP. Thus, even though HHA uses longer paths in presence of obstacles, it delivers packets on the shortest path in most cases which causes a small average APTS compared to that of FLD and DPP, although it supports delivery ratio as good as FLD. As with ADPP, TOP6 consistently shows lower APTS than TOP5 because of the use of smaller paths in many cases which again demonstrates the superiority of TOP6 over all other topologies.

To summarize, HHA supports delivery ratio as good as FLD with smaller number of transmissions on average compared to both FLD and DPP. The price to pay is a slight increase in average delay because of the increased length of the path used by HHA in the presence of obstacles. TOP6 again shows best delivery ratio with lower delay and transmissions than that of TOP5.

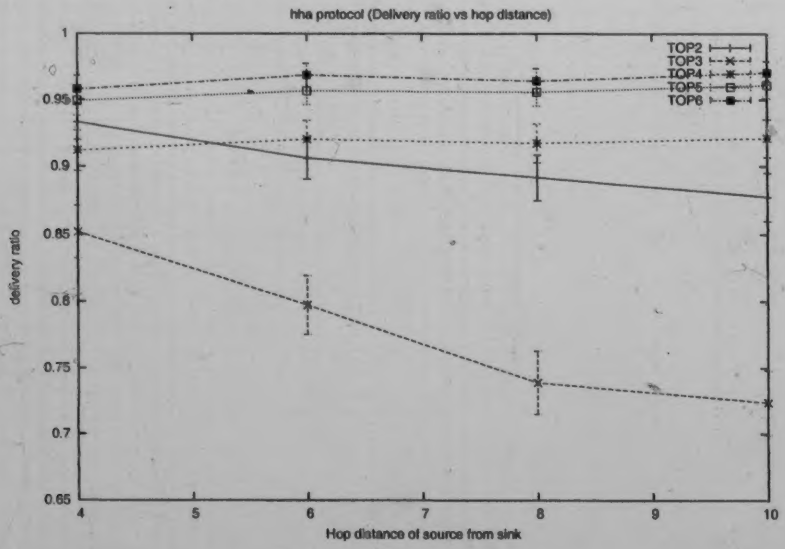

Figure 4.20: Delivery ratio vs hop distance with HHA. a = 20m, b = 4m, blob speed = 15cm/sec and TTL = 0.5 sec

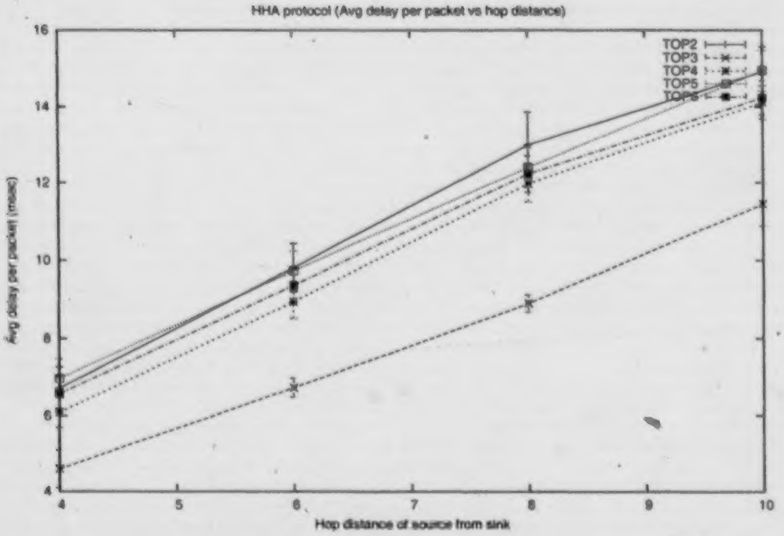

Figure 4.21: ADPP vs hop distance with HHA. a = 20m, b = 4m, blob speed = 15cm/sec and TTL = 0.5 sec

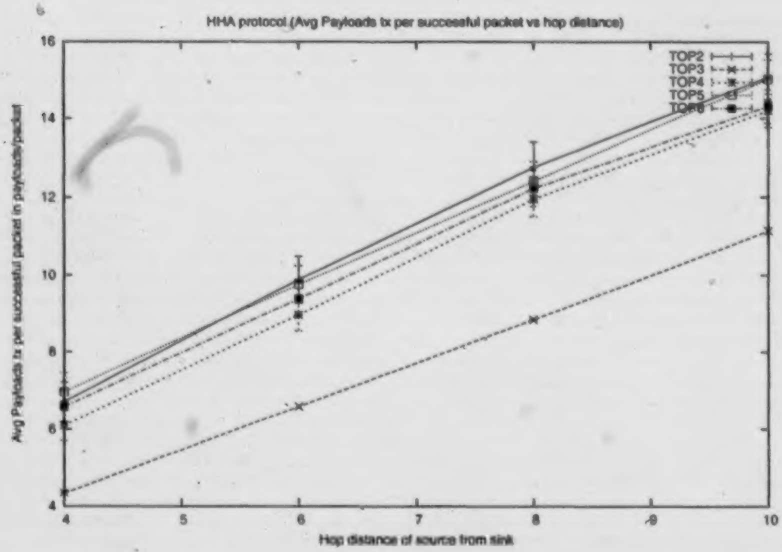

Figure 4.22: APTS vs hop distance with HHA. a = 20m, b = 4m, blob speed = 15cm/sec and TTL = 0.5 sec

	FLD	**DPP**	**HHA**
Delivery ratio	97%	90%	97%
Average Delay Per Packet (ADPP) in msec/packet	10.88	11.14	14.24
Average number of Payloads Transmitted per Successful packet (APTS) in payloads/packet	188.60	21.96	14.32

Table 4.2: Performance of FLD, DPP and HHA on TOP6 at a hop distance of 10 hops (a = 20m, b = 4m, speed = 15 cm/sec and TTL = 0.5 sec for HHA)

Experiment with Error Blob Size

In this experiment, we vary the size of the error blobs to see the effect of larger and smaller obstructions in the network under the Hop-by-Hop Acknowledgment with local update (HHA) protocol. We keep the semi-minor axis fixed at b = 4m and vary the length of semi-major axis a. In this experiment, we keep the hop distance of the source node from the sink fixed at 10 hops, the speed of each error blob fixed at 15 cm/sec and TTL fixed at 0.5 sec.

Figure 4.23 shows delivery ratio of HHA on different topologies against error blob size. As with our experiment on HHA with hop distance, delivery ratio with HHA for all topologies against error blob size remains closely similar to that with FLD (Figure 4.5) which is considerably higher than that with DPP (Figure 4.14). This is because HHA utilizes the underlying redundancies of the topologies like FLD, although in a different single-path approach. As with FLD, the delivery ratio of the best topology TOP6 remains as high as 95% with error blobs as large as a = 30m.

Figure 4.24 shows Average delay Per Packet (ADPP) of HHA on different topologies against error blob size. As with FLD, TOP3 has the lowest average delay which remains more or less constant with larger error blob sizes since packets are delivered on a single path and in presence of error blobs the packet is mostly dropped because of the small TTL. As with FLD (Figure 4.6), TOP2, TOP4, TOP5 and TOP6 all show increase in average delay with larger error blobs since larger blobs increase the probability of a link failure and re-computation of a new and long suboptimal path. As with FLD, this rate of increase goes up as we move from TOP2 to TOP4, from TOP4 to TOP5 and from TOP5 to TOP6.

In HHA, however, the difference in increase rate among different topologies is more acute than in FLD because unlike FLD, HHA computes alternative paths locally (at the point of failure) rather than globally (at the source) and local computation causes considerably longer paths. Thus, although TOP4, TOP5 and TOP6 has lower ADPP at smaller blob sizes (see Figure 4.24), their ADPPs exceed that of TOP2 at larger blob sizes. TOP6 has lower ADPP than that of TOP5 up to a blob size of a = 25m. After that, TOP6 has higher ADPP because it starts choosing paths through distant quadrants. Since error blobs as large as $a >$ 25m are unlikely, TOP6 remains the best candidate in terms of delay and delivery ratio. Table 4.3 summarizes the performance of three protocols applied on our best topology TOP6 with error blob size of a = 30m and b = 4m. As can be seen from the table, TOP6 at a = 30m has an ADPP of 17.40 msec with HHA compared to 11.26 msec with FLD and 11.27 msec with DPP. This suggests that if error blobs are very large, which is an unlikely situation, HHA shows poor delay characteristics with TOP6. A more intelligent multi-path scheme would probably be a better approach in such cases.

Figure 4.25 shows Average number of Payloads Transmitted per Successful packet (APTS) of HHA on different topologies against error blob size. As expected, APTS shows similar behavior as ADPP (Figure 4.24) since packets are delivered along a single path and both delay and number of transmissions are proportional to the length of the path. As can be seen from the summary in Table 4.3, we have an APTS of 17.33 payloads per packet for TOP6 with HHA at a = 30m compared to 184.24 payloads per packet with FLD and 21.41 payloads per packet with DPP which demonstrates the superiority of HHA protocol in terms of number of transmissions, even with very large error blobs.

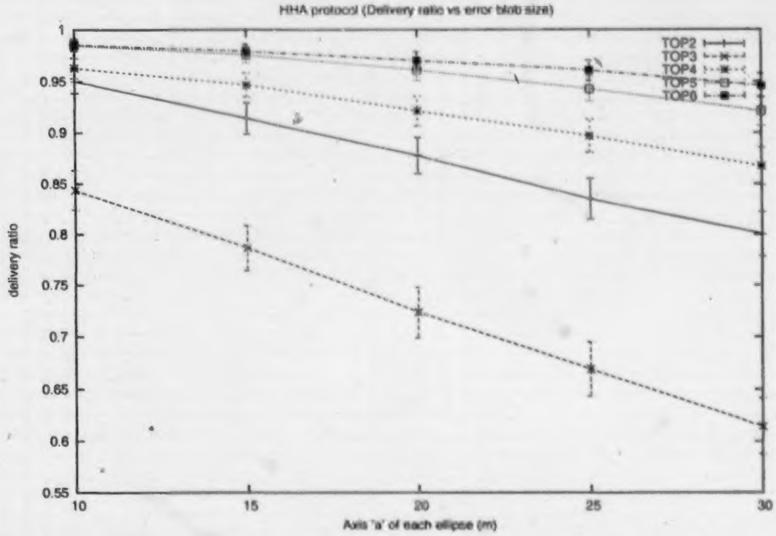

Figure 4.23: Delivery ratio vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec

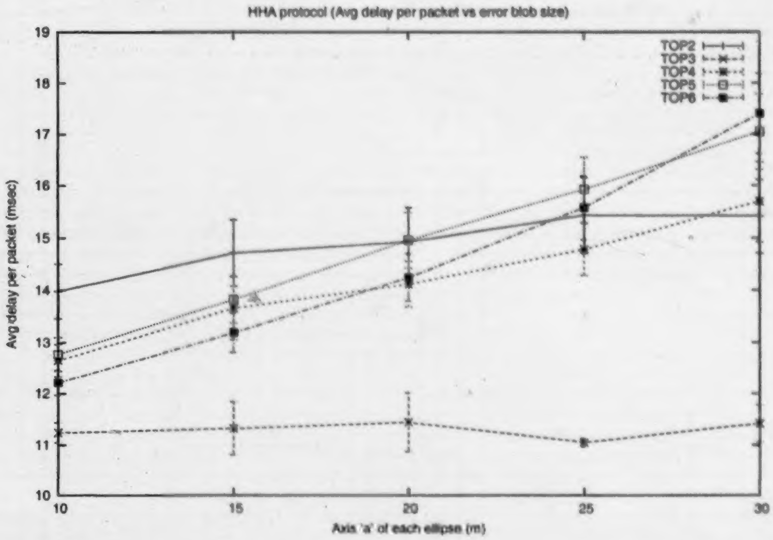

Figure 4.24: ADPP vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec

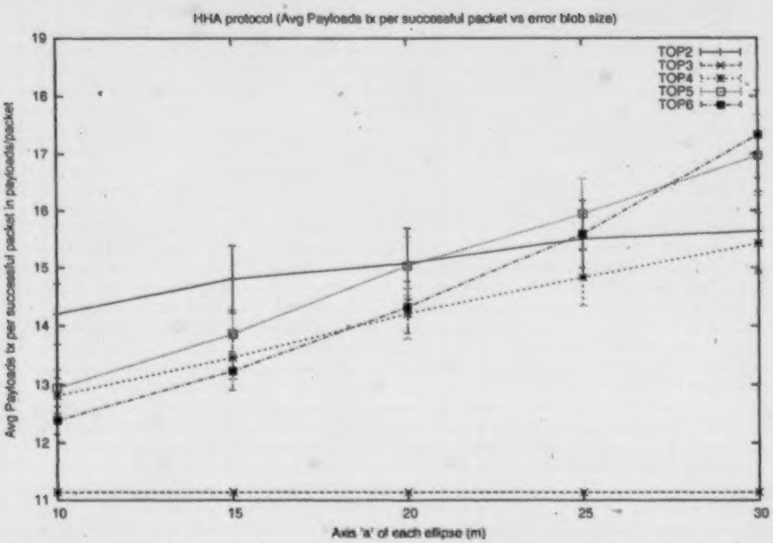

Figure 4.25: APTS vs error blob size with HHA. Hop distance = 10, blob speed = 15cm/sec and TTL = 0.5 sec

	FLD	**DPP**	**HHA**
Delivery ratio	95%	82%	96%
Average Delay Per Packet (ADPP) in msec/packet	11.26	11.27	17.40
Average number of Payloads Transmitted per Successful packet (APTS) in payloads/packet	184.24	21.41	17.33

Table 4.3: Performance of FLD, DPP and HHA on TOP6 with error blob dimension a = 30m, b = 4m (hop distance=10 hops, speed=15 cm/sec and TTL=0.5 sec for HHA)

4.6 Possible Effects with Multiple Sources

In our simulation, we use a single node as the packet generating source with all other nodes working merely as forwarding nodes. This has been done to keep our experiments simple since our main goal is to evaluate the resiliency of different protocols on our proposed topologies. Even with multiple sources, our protocols do not introduce MAC layer collisions since communication takes place on point-to-point dedicated links. However, multiple sources can introduce the problems of congestion, lack of buffer space and queuing delay. Note that our protocols do not handle the problem of load balancing. Thus, the problem of having a particular set of nodes/links carrying traffic most of the time with other nodes/links being idle is possible in our protocols. Examining this aspect of the protocols used is beyond the scope of this thesis.

With FLD and DPP, a node forwards a packet blindly (without considering the current link conditions) as soon as the node receives the packet and there is no anticipation for ACK. With just one source node and a low packet generation rate of 10 1-Kb packets a second, congestion never occurs and the forwarding node always finds the desired outgoing links free when it wants to forward a packet on these links. Therefore, there is no queuing delay and lack of buffer space. However, with very high packet generation rates or with multiple sources, possibly with different packet generation rates, congestion can build up, *i.e.*, a forwarding node may find its outgoing link(s) busy transmitting a previously received packet. In such cases, the node has to put the new packet on the SEND_BUFFER where the packet waits for the outgoing link to become free again. This introduces queuing delay and the possibility of lack of buffer space. Therefore, with multiple sources in FLD and DPP, we would have to design experiments to examine the effects of limited buffer space, the amount of buffer space needed for the best performance and the effects of queuing delay. Assuming infinite buffer space, we expect that even with multiple sources FLD and DPP would show similar relative behavior as with a single source in terms of delivery ratio and Average number of Payloads Transmitted per Successful packet (APTS) on average since packets would face the same forwarding logic, although sometimes delayed by the wait in buffer due to congestion, and the same degree of obstructions by the error blobs on average.

With HHA, packets are kept in the PACKET_BUFFER if no route to the sink can be found in presence of a nearby error blob(s). If more packets arrive when this is the case, they are kept in the buffer in first-in-first-out manner. Thus, with HHA, we have queuing delay and need for large buffer space even with a single packet generating source when error blobs are present. In our experiments, we keep the buffer size infinite and include queuing delays in the ADPP (average delay per packet) metric. It would be interesting to experiment with the size of the buffer and see the resulting effects. With multiple sources, congestion may build up and more queuing delay resulting from this congestion may arise, even in the absence of error blobs. As with FLD and DPP, we expect that with an assumption of infinite buffer space, HHA with multiple sources would show similar behavior as with a single source in terms of delivery ratio and APTS on average since packets would experience the same forwarding logic, although sometimes delayed by a longer period due to queuing delays resulting from congestion, and the same degree of obstructions by the error blobs on average. It would be interesting to design and experiment with a more intelligent version of HHA that routes packets around the areas of congestion and selects routes intelligently to evenly balance forwarding loads on nodes and links.

4.7 Chapter Summary

In this chapter, we have performed a dynamic evaluation of our proposed deployment topologies by simulating three simple routing protocols, FLD, DPP and HHA, on these topologies. These protocols attack the problems of link failures and resiliency from three different perspectives. We have presented the design of these protocols in detail and performed experiments with varying hop distance, error blob size, error blobs speed and TTL (for HHA) to evaluate and compare these protocols on different topologies in terms of packet delivery ratio, delay of delivery and number of transmissions incurred. Our results show that FLD achieves a very high degree of delivery ratio at the shortest delay but incurs excessive transmissions in the network. DPP fails to fully utilize the inherent redundancies of the topologies since it does not adjust its paths dynamically with current network conditions but it works as a representative of disjoint multi-path routing schemes. HHA achieves delivery ratio as good as FLD with a smaller average number of transmissions per packet than both FLD and DPP by using single but dynamically adjusted paths to deliver packets. The price to pay is a slightly longer average delay per packet, especially with very large error blobs. Our results also justify the addition of two extra links on TOP5 to produce TOP6 since TOP6 not only supports higher delivery ratio by providing more redundancy but also reduces the average delay by providing shorter alternative paths to the sink in some cases.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

In this thesis, we have designed robust two-dimensional grid-based deployment schemes for underwater sensor networks that use point-to-point optical communication links. In particular, we have considered sensor nodes deployed in a grid structure and then designed schemes to select point-to-point optical links between adjacent nodes to generate robust deployment topologies. Our design goals have been to include redundant paths in the topology to improve robustness in the face of link failures and to include short paths from sink to each sensor node to support low-cost communication in typical failure-free environment. The trade-off we have made while doing so is to keep the number of total and per node communication interfaces as small as possible in order to reduce the cost of deployment.

We have considered three cases where each node in the grid is constrained to have no more than 1, 2 and 3 interfaces. For maximum 1 and 2 interfaces per node cases, we have proposed deployment topologies consisting of four Hamiltonian cycles in the network. For maximum 3 interfaces per node case, we have designed shortest-path tree based deployment topologies where redundant links have been added at strategic points in the network on top of a shortest path tree in order to improve the robustness while at the same time allowing shortest path communication in the network in the absence of failures. To this end, we have designed a formulation pattern for a 3-degree constrained shortest path tree in a grid rooted at the sink and spanning all nodes in the grid with (LB+2) number of 3-degree nodes in the worst case where LB is the lower bound on the number of 3-degree nodes in such a tree. Our proposed formulation pattern works for any grid dimension and any placement of the sink inside the grid. With this shortest path tree as the base, we have designed a series of deployment topologies that offer higher degrees of robustness by adding additional links on top of the shortest path tree at strategic points.

We have designed 2-edge-connected deployment topologies in order to support deterministic robustness whenever our constraint on the maximum number of interfaces a node can have allowed us to do so. In order to examine the probabilistic robustness of our proposed deployment topologies, we have simulated our topologies with isolated and patterned failure models under static settings. Our results demonstrate that our best topology TOP6 offers a very high degree of robustness even though it has a small number of optical links compared to the entire potential grid graph.

In order to examine the dynamic behavior of our proposed topologies, we have performed a detailed dynamic evaluation of these topologies by simulating them with three simple resilient routing protocols. We have selected three simple routing protocols that approach the problem of resiliency from three different perspectives. We have used a single packet generating source node and multiple moving obstacles in the network and evaluated the performance of the three protocols applied on our topologies in terms of packet delivery ratio (resiliency), average delay of delivery of packets to the sink and overall communication overhead. Our results show that the HHA (Hop-by-Hop Acknowledgment with local update) protocol supports resiliency as good as flooding with fewer number of transmissions on average compared to both flooding and multi-path based DPP (Dual Paths Protocol) by using a single but dynamically adjusted path to deliver packets. However, HHA incurs slightly higher delay of delivery on average compared to the other protocols since it uses local updates that occasionally result in longer paths compared to flooding and DPP. Our results also show that our best topology TOP6 outperforms all other topologies in terms of resiliency. In addition, it has lower average delay than that of TOP5 since it not only has higher degree of redundancy but also has shorter alternate paths to the sink from many nodes. This justifies the addition of two more links on TOP5 to generate TOP6.

5.2 Future Directions for Research

We have designed a formulation pattern for a 3-degree constrained shortest path spanning tree in a grid graph of arbitrary dimension rooted at an arbitrary grid node (the sink) that has (LB+2) 3-degree nodes in the worst case where LB is the lower bound on the number of 3-degree nodes in such a tree. It would be interesting to design a formulation pattern for such a tree for which the number of 3-degree node is equal to this lower bound or prove that such a tree cannot be formed. If the latter is the case, it would also be interesting to prove or disprove that our pattern is a 3-degree constrained shortest path spanning tree rooted at the sink with "minimum" number of 3-degree nodes.

Another interesting future work would be to extend our results to three-dimensional deployment scenarios. For example, we can consider a three-dimensional grid instead of a 2D grid and see if our proposed patterns can be extended to the 3D case. One approach could be to divide the 3D grid into horizontal 2D planar grids and then apply our patterns on these 2D grids and add additional links to provide connectivity between adjacent 2D grids to come up with a connected 3D deployment. Another approach could be to introduce new constraints on node degrees, *e.g.*, maximum 4 or 5 interfaces per node, and then redesign the deployment pattern in a three-dimensional approach, *e.g.*, divide the entire 3D grid into eight 3D octants (in contrast to four 2D quadrants) and try to formulate degree constrained shortest path spanning tree for each octant. It would also be interesting to remove the assumption of grid-based deployment and consider other deployment schemes such as triangular or hexagonal deployment.

In our dynamic evaluation, we have used a single packet generating source in the entire network with all other nodes working merely as forwarding nodes. While this setting shows us the degree of resiliency of the routing scheme applied on the underlying topology, having multiple sources in the network introduces issues such as congestion, queuing delays and lack of buffer space. Therefore, it would be worthwhile to investigate these aspects of performance by having multiple sources in the network, possibly with different packet generation rates for different sources. Also, we have ignored the problem of balancing packet forwarding loads evenly among nodes in our design of the routing protocols. An interesting future work would be to design routing protocols that select routes more intelligently to evenly balance forwarding loads on different nodes in the network.

The multi-path based Dual Paths Protocol (DPP) that we have used in our dynamic evaluation selects the two routing paths at the start of network operation and never adjusts these paths later on in order to improve the percentage of successful delivery. It would be interesting to use a more intelligent and dynamic multi-path protocol [48, 30, 50] on our topologies to examine the performance of such protocols on these topologies.

The single-path based Hop-by-Hop Acknowledgment with local update (HHA) protocol includes routing information in the packet which increases the number of bits that need to be transmitted. In contrast, we could design a version of HHA where each node takes routing decision independently to avoid the inclusion of routing paths in the packets. However, this would increase the amount of computation in the network since each subsequent node would have to perform the re-computation of routes in the face of link failures instead of just the node that detects the failure. It would be interesting to examine this communication-computation trade-off resulting from such decisions. In our HHA protocol, a node detecting a failure has to include this information in each subsequent packet since the subsequent nodes do not remember this information. It would be interesting to design a more intelligent version of HHA in which a node remembers link failure information sent by the preceding nodes and makes use of this information intelligently so that the preceding nodes do not have to transmit this information more than once.

Bibliography

[1] I.F. Akyildiz, D. Pompili, and T. Melodia. Challenges for efficient communication in underwater acoustic sensor networks. *ACM Mobile Computing and Communication Review*, July 2007.

[2] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *Proc. of MobiHoc '06*, Florence, Italy, May 2006.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proc. of the Seventh International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001.

[4] R.R. Choudhury and N.H. Vaidya. Ad hoc routing using directional antennas. *Technical Report, UIUC*, August 2002.

[5] R.R. Choudhury, X. Yang, R. Ramanathan, and N.H. Vaidya. Using directional antennas for medium access control in ad hoc networks. In *Proc. of MOBICOM '02*, Atlanta, GA, September 2002.

[6] F. Dai, Q. Dai, and J. Wu. Power efficient routing trees for ad hoc wireless networks using directional antenna. *Ad Hoc Networks 3 (2005)*, pages 621–628.

[7] F. Dai and J. Wu. Efficient broadcasting in ad hoc wireless networks using directional antennas. *IEEE Transactions on Parallel and Distributed Systems*, 17(4), April 2006.

[8] J. Diaz, J. Petit, and M. Serna. Random scale sector graphs. *Tecnical Report, Universitat Politecnica Catalunya*, 2003.

[9] P.G. Doyle and J.L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, Washignton D.C., 1984.

[10] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *International Conference on Acoustics, Speech and Signal Processing*, Salk Lake City, UT, May 2001.

[11] N. Farr, A.D. Chave, L. Freitag, J. Preisig, S.N. White, D. Yoerger, and F. Sonnichsen. Optical modem technology for seafloor observatories. In *Proc. of IEEE OCEANS'06*, Boston, MA, September 2006.

[12] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.

[13] J.W. Giles and I.N. Bankman. Underwater optical communications systems part 2: basic design considerations. In *Proc. of IEEE MILCOM 2005*, Atlantic City, NJ, October 2005.

[14] H. Gupta, S.R. Das, and Q. Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. In *Proc. of MobiHoc '03*, Annapolis, MD, June 2003.

[15] P.C. Gurumohan and J. Hui. Topology design for free space optical networks. In *Proc. of ICCCN 2003*, Dallas, TX, October 2003.

[16] Y.T. Hou, Y. Shi, J. Pan, S.F. Midkiff, and K. Sohraby. Single-beam flow routing for wireless sensor networks. In *Proc. of IEEE GLOBECOM '05*, St. Louis, MO, November 2005.

[17] C. Hu, Y. Hong, and J. Hou. On mitigating the broadcast storm problem with directional antennas. In *Proc. of IEEE International Conference on Communication*, Anchorage, Alaska, May 2003.

[18] A.F. Harris III and M. Zorzi. On the design of energy-efficient routing protocols in underwater networks. In *Proc. of IEEE SECON 2007*, San Diego, CA, June 2007.

[19] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley and Sons, 1991.

[20] P. Karn. Maca a new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, 1990.

[21] A. Kashyap, K. Lee, M. Kalantari, S. Khuller, and M. Shayman. Integrated topology control and routing in wireless optical mesh networks. *Computer Networks Journal*, 51:4237–4251, October 2007.

[22] D. Kedar and S. Arnon. A distributed sensor system for detection of contaminants in the ocean. In *Proc of SPIE (Society of Photographic Instrumentation Engineers) Vol. 6399, 639903(2006)*, Stockholm, Sweden, 2006.

[23] B. Kershner. The number of circles covering a set. *American Journal of Mathematics, 61:665-671*, 1939.

[24] S. Khuller, K. Lee, and M. Shayman. On degree constrained shortest paths. In *European Symposium on Algorithms*, Eivissa, Spain, October 2005.

[25] E.F. Krause. *Taxicab Geometry*. Dover, NY, 1987.

[26] M. Krishnamoorthy, A.T. Ernst, and Y.M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, 7(6), November 2001.

[27] M. Kubisch, H. Karl, and A. Wolisz. A mac protocol for wireless sensor networks with multiple selectable, fixed-orientation antennas. *Technical Report, Technische Universitat Berlin*, February 2004.

[28] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis, 2nd Edition.* McGraw-Hill Inc, 1991.

[29] U. Lee, J. Kong, J.-S. Park, E. Magistretti, and M. Gerla. Time-critical underwater sensor diffusion with no proactive and negligible reactive floods. In *Proc. of ISCC'06*, Sardinia, Italy, June 2006.

[30] S. Li and Z. Wu. Node-disjoint parallel multi-path routing in wireless sensor networks. In *In Proc. of the Second International Conference on Embedded Software and Systems (ICESS '05)*, Xian, China, December 2005.

[31] J. Llorca, A. Desai, and S. Milner. Obscuration minimization in dynamic free space optical networks through topology control. In *Proc. of IEEE MILCOM 2004*, Monterey, CA, November 2004.

[32] M. Molins and M. Stojanovic. Slotted fama: a mac protocol for underwater acoustic networks. In *Proc. of IEEE OCEANS'06*, Boston, MA, September 2006.

[33] U.N. Okorafor and D. Kundur. Efficient routing protocols for a free space optical sensor network. In *Proc. of IEEE MASS 2005*, Washington D.C., November 2005.

[34] B. Peleato and M. Stojanovic. A mac protocol for ad-hoc underwater acoustic sensor networks. In *Proc. of WUWNet'06*, Los Angeles, CA, September 2006.

[35] D. Pompili, T. Melodia, and I.F. Akyildiz. Deployment analysis in underwater acoustic wireless sensor networks. In *Proc. of WUWNet'06*, Los Angeles, CA, September 2006.

[36] D. Pompili, T. Melodia, and I.F. Akyildiz. A resilient routing algorithm for long-term applications in underwater sensor networks. In *Proc. of Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, Lipari, Italy, June 2006.

[37] D. Pompili, T. Melodia, and I.F. Akyildiz. Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks. In *MobiCom'06*, Los Angeles, CA, September 2006.

[38] F. Schill, U.R. Zimmer, and J. Trumpf. Visible spectrum optical communication and distance sensing for underwater applications. In *Proc. of the Australasian Conference on Robotics and Automation*, Canberra, Australia, December 2004.

[39] W.K.G. Seah and H.-X. Tan. Multipath virtual sink architecture for underwater sensor networks. In *Proc. of the MTS/IEEE OCEANS2006 Asia Pacific Conference*, Singapore, May 2006.

[40] W.K.G. Seah and H.P. Tan. Multipath virtual sink architecture for wireless sensor networks in harsh environments. In *Proc. of InterSense'06*, Nice, France, May 2006.

[41] S. Skeina. *Implementing Discrete Mathematics: combinatorics and Graph Theory with Mathematica.* Addison-Wesley, Redwood City, CA, 1990.

[42] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *Proc. of IEEE International Conference on Communication*, Helsinki, Finland, June 2001.

[43] J. H. Smart. Underwater optical communications systems part 1: variability of water optical parameters. In *Proc. of IEEE MILCOM 2005*, Atlantic City, NJ, October 2005.

[44] P. Sun, W.K.G. Seah, and P.W.Q. Lee. Efficient data delivery with packet cloning for underwater sensor networks. In *Proc. of UT07 and SSC07*, Tokyo, Japan, April 2007.

[45] A.S. Tanenbaum. *Computer Networks, 4th Edition.* Prentice Hall PTR, New Jersey, 2002.

[46] Victoria Experimental Network Under the Sea. http://www.venus.uvic.ca/.

[47] D. Tian and N.D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proc. of First International Workshop of Wireless Sensor Networks and Applications (WSNA '02)*, Atlanta, GA, September 2002.

[48] J. Tsai and T. Moors. A review of multipath routing protocols: from wireless ad hoc to mesh networks. In *Proc. ACoRN Early Career Researcher Workshop on Wireless Multihop Networking*, July 2006.

[49] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage and retrieval with an underwater sensor network. In *ACM SenSys'05*, San Diego, CA, November 2005.

[50] S. Waharte and R. Boutaba. Totally disjoint multipath routing in multihop wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC 2006)*, Istanbul, Turkey, June 2006.

[51] W.-C. Wang, C.-C. Hu, and Y.-C. Tseng. Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks. In *Proc. of the First International Conference on Wireless Internet (WICON '05)*, Budapest, Hungary, July 2005.

[52] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proc. of SenSys '03*, Los Angeles, CA, November 2003.

[53] D.B. West. *Introduction to Graph Theory, 2nd Edition.* Prentice Hall, New Jersey, 2006.

[54] J. Wu and F. Dai. A distributed formation of virtual backbone in manets using adjustable transmission ranges. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS '04)*, Tokyo, Japan, March 2004.

[55] J. Wu, F. Dai, M. Gao, and I. Stojmenovic. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. *Journal of Communications and Networks*, 4(1), March 2002.

[56] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad-hoc wireless networks. In *Proc. of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, WA, 1999.

[57] P. Xie and J.H. Cui. R-mac: An energy-efficient mac protocol for underwater sensor networks. In *Proc. of WASA 2007*, Chicago, IL, August 2007.

[58] P. Xie, J.H. Cui, and L. Lao. Vbf: vector-based forwarding protocol for underwater sensor networks. *UCONN Technical Report UbiNet-TR05-03*, February 2005.

[59] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for adhoc routing. In *Proc. of the Seventh International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001.

[60] F. Ye, G. Zhong, S. Lu, and L. Zhang. Energy-efficient robust sensing coverage in large sensor networks. *UCLA Technical Report*, 2002.

[61] H. Zhang and J. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *UIUC Technical Report*, March 2005.

Appendix A

Implementation of FLD, DPP and HHA

In this appendix, we present the implementation details of the three routing protocols that we have used in our analysis.

A.1 Implementation of Flooding (FLD)

In flooding (FLD), each node has one separate first-in-first-out SEND_BUFFER for each outgoing link. In our simulation, we set the capacity of this buffer large enough so that an overflow never occurs. Whenever a node receives a packet on one of its incoming link, it checks the SOURCE and SEQ field of the packet and looks up its FLOODING_TABLE (described shortly) to see if this packet has been seen before. If so, the packet is simply ignored. If not, the source and sequence number of the packet is inserted into FLOODING_TABLE to avoid forwarding the packet again in future. Then the node finds out which of its neighbors the packet arrived from, let us call this neighbor n_1. The receiving node then inserts this packet to the SEND_BUFFER of each of its outgoing links except the one that leads to n_1. Packets are transmitted one by one from the head/front of each SEND_BUFFER on the corresponding outgoing link.

FLOODING_TABLE in a node is the data structure that records which packets (sequence number) from which source node have been seen and forwarded before. It consists of N linked lists where N is the number of nodes in the network. The list for node i contains all the sequence numbers that have been seen by the node in question so far. The sequence numbers in a list are maintained in descending order of magnitude in order to make insertion and lookup very fast, since a node most of the time sees a sequence number from a source that is close to the largest sequence number seen so far from that source.

The simulator of FLD has three events: PACKET_GENERATED, PACKET_ARRIVED and BLOBS_MOVE. A PACKET_GENERATED event occurs at node i when node i has generated a data packet and it takes place ten times a second at a source node in our simulation. A PACKET_ARRIVED event occurs at node i when node i has received a packet on one of its incoming links. A BLOBS_MOVE event for error blob j, $1 \leq j \leq 3$, occurs when blob j needs to move in a random direction by 1 decimeter to achieve its desired speed. Pseudo-code of the event-handling procedures of these events have been presented in Appendix C.1.

Events are maintained in a priority queue and scheduling an event means inserting the event in this queue. Extracting an event from this priority queue returns the event with the earliest time of occurrence. The main simulation loop extracts one event from the queue at each iteration, advances the simulation clock to this event's time of occurrence and performs the event-specific tasks (calls the appropriate event-handling procedure) depending on the type of the event.

A.2 Implementation of Dual Paths Protocol (DPP)

The dual paths protocol (DPP) is implemented in the same way as FLD except the following differences. Nodes no longer need to keep the FLOODING_TABLE data structure except the sink which needs it in order to avoid receiving duplicate packets. The forwarding technique is slightly different in DPP than in FLD which calls for appropriate changes. At the beginning of network operation, each node computes the two paths suggested by DPP (one path for TOP1) from each node in the network to the sink and finds out for which source node it should forward a packet to which of its outgoing links. It stores this information in a data structure called FORWARDING_TABLE. Because of the disjoint nature of the selected paths, a node should forward a packet on a single outgoing link if the source of the packet is a node different than itself. For a packet that is generated in the forwarding node itself, the node should forward the packet on two different outgoing links (except in TOP1). During network operations, a forwarding node can find out which outgoing link(s) it should forward a packet on by simply looking up its FORWARDING_TABLE with the source of the packet.

The simulator for DPP has the same three events as FLD: PACKET_GENERATED, PACKET_ARRIVED and BLOBS_MOVE. The procedure for BLOBS_MOVE event is exactly the same as in FLD. For PACKET_GENERATED and PACKET_ARRIVED events, DPP works the same way as FLD except that instead of placing the packet blindly on all SEND_BUFFERs, the node in question looks up its FORWARDING_TABLE with the source of the packet and places the packet only on the SEND_BUFFER(s) as suggested by the lookup and schedules appropriate PACKET_ARRIVED events.

A.3 Implementation of Hop-by-Hop Acknowledgment Protocol (HHA)

In hop-by-hop acknowledgment with local update protocol (HHA), each node has a first-in-first out buffer called PACKET_BUFFER, as mentioned in Section 4.1.3, where all generated and arrived data packets are kept and packets are forwarded from this buffer in first-in-first-out manner the logic of which has been described in Section 4.1.3. In our simulation, we keep this buffer large enough so that overflow never occurs. For each outgoing link of a node, we can have maximum two packets waiting to be transmitted at the same time: one is a DATA packet that the node decides to forward on this link and the other is an ACK packet that the node needs to forward on this link in response to a DATA packet arrival. Thus, for each outgoing link, we have a first-in-first out SEND_BUFFER that can hold maximum two packets at a certain moment. Whenever a node decides to transmit a packet on an outgoing link, it inserts the packet on the SEND_BUFFER of the corresponding link. Packets are physically transmitted from the SEND_BUFFER in first-in-first-out fashion. Note that once a node places a DATA packet on the SEND_BUFFER of an outgoing link for transmission, it does not attempt the transmission of another DATA packet on the same outgoing link until the packet is completely transmitted and moved out of the buffer and either an ACK for that packet is received or a time ACK_INTERVAL is passed after the transmission. Also, once a node places an ACK packet on the SEND_BUFFER of an outgoing link for transmission, it does not attempt the transmission of another ACK packet on the same outgoing link until the ACK packet is completely transmitted and the node at the other end of the link transmits another DATA packet in response to this ACK. As a result, a SEND_BUFFER can have at most two packets in it at a particular point in time, one DATA and one ACK.

A node considers one of its outgoing links to be down when it transmits a DATA packet on this link and does not receive an ACK before the timeout occurs. To detect the revival of a down link, periodic transmission of echo packets or a timer can be used. It our implementation, we assume that a node automatically knows when a link comes back up, perhaps with the help of a special-purpose hardware.

The simulator for HHA has total five events, three of which are the same as FLD and DPP: PACKET_GENERATED, PACKET_ARRIVED and BLOBS_MOVE. The two new events in HHA are ACK_TIMEOUT and RETRY_TIMEOUT. An ACK_TIMEOUT event at node i indicates that a time equal to ACK_INTERVAL has passed since the transmission of a data to a neighbor, say node j, and no ACK from node j has been received. A RETY_TIMEOUT event at node i indicates that a time equal to RETRY_INTERVAL has passed since node i tried to forward a packet but could not do it since there was no active outgoing link from it or no path from it to the sink was found. Pseudo-codes of the event-handling procedures of these events have been presented in Appendix C.2.

A.4 Packet Formats

FLD and DPP both have only one type of packet: fixed-length data packets. The format and size of different fields of such a packet is shown in Table A.1. We number the nodes (total 144) from 0 to 143 and use these numbers as node IDs. Since our highest node ID is 143, we use 8 bit fields for denoting node IDs, *e.g.*, SOURCE and DESTINATION fields in Table A.1. The TYPE field is 3 bits and it indicates what type of packet we are dealing with (can only be DATA for FLD and DPP). The SEQ field carries sequence number of the packet which is generated by the source of the packet. GENERATION_TIME is a time-stamp that indicates when the packet was generated (inserted by the source) which a forwarding node can compare against current time to determine whether the packet is too old. The total size of each DATA packet in FLD and DPP is 1.051 Kb.

With HHA, we have variable length DATA packets and fixed length ACK packets. The packet format of DATA packet remains the same as with DPP and FLD (see Table A.1) except that variable length REMAINING_PATH and LINKS_DOWN fields are added at the end. Since each node ID takes 8 bits, if there are n nodes in the path, the REMAINING_PATH field will be $n * 8$ bits in length. On the other hand, each link is denoted by two node IDs. Thus, if there are m links in the LINKS_DOWN field, its size will be $m * 8 * 2$ bits. Additionally, there are two fields, each 8 bits, indicating the length of these two fields (n and m).

The ACK packet in HHA has three fields: TYPE, SOURCE and SEQ. Thus, the total size is 27 bits ($3 + 8 + 16$). The SOURCE and SEQ fields are used by the node receiving the ACK packet to match the ACK packet with outstanding data packets.

Field Name	**Size**
PAYLOAD	1 Kb
TYPE	3 bits
SOURCE	8 bits
DESTINATION	8 bits
SEQ	16 bits
GENERATION_TIME	16 bits
Total	1.051 Kb

Table A.1: Different fields and their sizes in a fixed-length DATA packet of FLD/DPP protocol

Appendix B

Verification and Validation of Simulation Model

In this appendix, we present the steps we have taken to perform the verification and validation of the simulation model described in Chapter 4. We have carried out a number of standard verification and validation techniques [19] on our simulation implementation and results. The first and the most important one is the modular design we have used throughout our implementation in order to keep it tractable and well-organized. Figure B.1 shows the key modules of our implementation and their relationships. We have implemented and tested each of these modules separately. The MAIN module is the starting point of the simulation and contains the logic for different experiments, *e.g.*, varying hop distance or varying error blob size. It uses the module TOPOLOGIES to generate the target topology and then invokes one of the three modules FLD, DPP and HHA which are responsible for performing simulation of FLD, DPP and HHA protocols, respectively.

The structure of each of the modules FLD, DPP and HHA is essentially the same with different logic to reflect the protocol in hand. Each of these modules contains an initialization routine, the routine with the main simulation loop and event handler routines for each event. Since these modules are similar in structure, Figure B.1 shows only the helper modules of FLD. The BFS module has the logic and data structure to run breadth first search (BFS) on a given graph and it uses the module QUEUE that provides logic and data structure to implement a queue that is required by the BFS algorithm. The BFS module is used to find out shortest paths in the graph and is useful in DPP and HHA protocols. The module RANDOM has routines that supply uniform random numbers and integers and routines that initialize the random number generator with the desired seed. The module ERROR_BLOBS has routines that place error blobs randomly on the grid and move the error blobs using random walk. It also has routines that verify whether or not a given link (or node) is inside an error blob or intersects an error blob. These routines are implemented with the support of the module ELLIPSE which has routines to perform ellipse-specific calculations since error blobs are elliptical in shape. The module STATION has routines that initialize and update different data structures of a station (*e.g.*, flooding table, buffers) and search for specific entries in these data structures (*e.g.*, search for a sequence number in the flooding table). Finally, the module TIMING maintains the event list and has routines for inserting and retrieving events from the event list which is maintained as a priority queue.

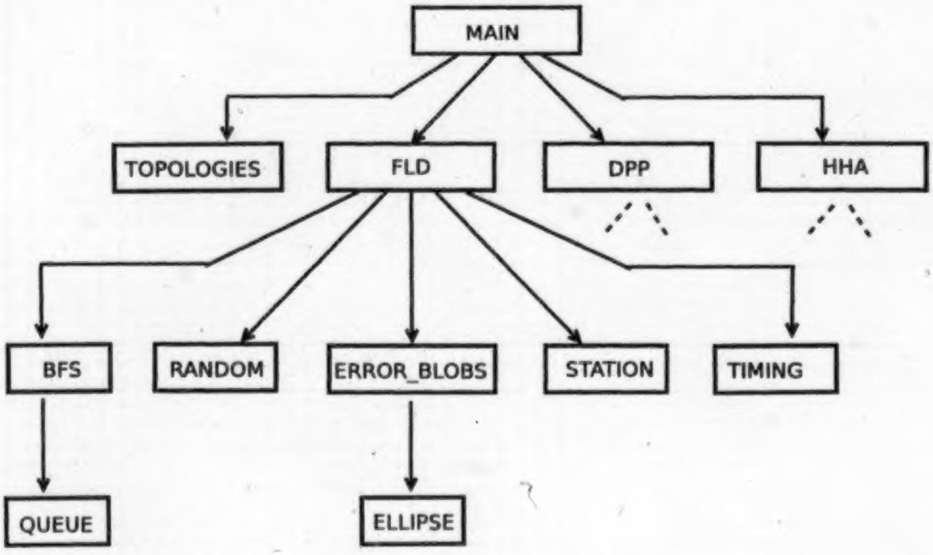

Figure B.1: Modules in the simulation implementation

The next technique after modular design that we have used to verify our implementation is anti-bugging [19]. We have placed codes at different places in our implementation to check for bugs and inconsistencies and exit from the simulation with error message if a bug or inconsistency is detected. For example, if a PACKET_ARRIVED event happens at a node, the receive buffer of the node must have a valid packet. We have assumed infinite buffer space in our model. Thus, whenever we need to insert a packet in a buffer, the buffer must have sufficient space to accommodate this packet. The number of dropped packets added with the number of delivered packets should be equal to the number of generated packets. In HHA, if a node other than the sink receives a DATA packet, the packet's REMAINING_PATH must be non-empty and indicate the next node the packet should be forwarded to. We have used anti-bugging code in our simulation to detect these and many other potential bugs and inconsistencies.

We have generated detailed traces of our implementation to check for errors and inconsistencies. We have run the simulation with simple settings and short duration and examined the generated traces to verify the model. In order to perform structured walk-through [19], we have documented detailed pseudo-codes of our implementations and examined these codes for errors and inconsistencies. We have also explained the pseudo-code of flooding (FLD) protocol to Dr. Janelle Harms in order to cross-check the correctness of our implementation. We have used visual verification whenever possible. For example, we have drawn the generated topologies from within the program to see that they are actually the topologies we have intended to use. To verify that the ellipse calculations are correct, we have drawn ellipses at different grid locations with different orientations and then we have drawn straight lines at different locations to see of they intersect or are inside the ellipse and if our program detects it correctly.

We have also run our implementation for some simple and tractable but representative cases to verify correctness by comparing the outputs with analytical estimates. We present one such example here. We use the settings of Figure 4.1 in Chapter 4 where node 6 sends a packet to the sink (node 67) using HHA protocol on TOP6. We use an error blob smaller than the one shown in Figure 4.1 so that the blob blocks only link 31-43 instead of both 31-43 and 32-44 as shown in the figure. We place an error blob (ellipse) with dimension $a =$ 15m and $b =$ 4m with horizontal orientation as shown in Figure 4.1. This error blobs blocks link 31-43 (not link 32-44) so that the packets first travels along the path 6-7-19-31 and then along the path 31-19-7-8-20-32-44-56-68-67 to reach the sink. We set the simulation time equal to 100 msec which is the time required to generate one packet. In our implementation, the simulation continues even after the specified time has expired so that all the packets generated so far are handled (delivered or dropped). With 100 msec simulation time, the simulation generates one packet and continues to run until this packet is delivered to the sink or dropped.

With this settings, let us calculate the amount of data transmitted to deliver this packet using HHA on TOP6. The initial intended path for this packet is 6-7-19-31-43-55-67 which has 6 hops although the packet is rerouted after it has traveled 3 hops. Once the packet faces the error blob at node 31, it is rerouted and delivered on the path 31-19-7-8-20-32-44-56-68-67 which is 9 hops in length. Each DATA packet has a fixed length portion of 1.067 Kb (a 1.051 Kb portion identical to a DATA packet in FLD and DPP, as shown in Table A.1, and two 8 bit fields to indicate the lengths of REMAINING_PATH and LINKS_DOWN fields). Since the packet travels total 12 hops (3 + 9), total data transmitted for the fixed length portions of DATA packets is 12.804 Kb (1.067 * 12). For each node in the REMAINING_PATH field, we have to transmit 8 bits of data at each hop. Thus, total data transmitted for REMAINING_PATH fields is 8 * (6 + 5 + 4) bits for the initial path and 8 * (9 + 8 + 7 + ... + 1) bits for the rerouted path which sum up to 0.48 Kb. For each link in the LINKS_DOWN field, we have to transmit 16 bits at each hop. The LINKS_DOWN field is empty throughout the initial path and has one link (31-43) throughout the recomputed path. Thus, total data transmitted for LINKS_DOWN fields is 16*9 bits or 0.144 Kb. Finally, ACK packets are sent at each 3 hops in the first path and each 9 hops in the rerouted path which results in a transmission of 324 bits or 0.324 Kb (12 * 27 since each ACK packet has 27 bits). Therefore, total data transmitted for the delivery of this packet is 13.752 Kb (12.804 + 0.48 + 0.144 + 0.324).

Since only one packet is generated and it is delivered, the delivery ratio should be 1. Since the size of the payload field is 1 Kb, Average number of Payloads Transmitted per Successful packet (APTS) should be 13.752 payloads/packet. We have observed that our implementation produces these exact same values.

In order to validate our simulation outputs, we have examined the outputs for consistency thoroughly in our analysis in Section 4.5. For example, flooding (FLD) and dual paths protocol (DDP) should have exactly same outputs for TOP2 since DPP uses two best paths from each source to the sink and TOP2 has exactly two paths from each source to sink which makes FLD work identically as DPP. In the plots shown in Section 4.5, we have seen that this is exactly the case. Another example is TOP3 which is a shortest path tree. With FLD and DPP, we should have the same delivery ratio and average delay per packet (ADPP) for TOP3 since there is exactly one path from each source to sink. However, because of unnecessary flooding in other branches of the tree, FLD incurs significantly more transmissions than DPP and thus has higher values of APTS. We have seen exactly this behavior in out plots in Section 4.5. With HHA on TOP3, packets are delivered on the exact same path as with FLD and DPP, although waiting at a stuck node can introduce some extra delay. Thus, Average number of Payloads Transmitted per Successful packet (APTS) should be the same with HHA as with FLD and DPP. However, because of longer and variable packet headers and the transmissions of ACK packets in HHA, we expect a slightly higher APTS with HHA on TOP3 and this is exactly what we have observed in the plots of Section 4.5.

Appendix C

Pseudo-code of Event Handlers for the Simulation Models

In this appendix, we present pseudo-codes for different event handlers in the simulation model for the flooding (FLD) and Hop-by-Hop Acknowledgment with local update (HHA) protocol. It has been assumed that there is a global variable named *sim_clock* that keeps track of the current simulation time. Also, an event has five different fields. The field *type* indicates what type of event it is, *e.g.*, PACKET_ARRIVED, PACKET_GENERATED etc. The field *time* indicates the time of occurrence of this event. The fields *node* and *link* indicate the node and the link, respectively, where the event has occurred. For BLOBS_MOVE event, the *node* field indicates the error blob which needs to move to a new location. Finally, The field *success* indicates whether the transmission of the packet was successful (relevant only for PACKET_ARRIVED event).

For HHA, we additionally assume that a node has a variable *forwarded_on_original_link* that indicates whether the packet at the head of the PACKET_BUFFER that is waiting for an ACK was transmitted on the link indicated by the original REMAINING_PATH field of the packet (a value of 1 for the variable) or it was forwarded on a link suggested by the recomputed path since the original intended link was down (a value of 0 for the variable). We also assume that a DATA packet in HHA has a new filed called *tx_end_time* that indicates the time when the transmission of this packet was finished. The *tx_end_time* field in an ACK packet indicates the time when the transmission of the corresponding DATA packet was finished. Therefore, when an ACK is transmitted, we copy the *tx_end_time* field of the corresponding data packet into the *tx_end_time* field of the ACK packet. This field indicates us when the timer for an ACK needs to be started.

The event handlers in HHA use the helper procedures FORWARD_ONE_PACKET and RECOMPUTE_PATH, each called with node i to indicate at which node the forwarding or re-computation should take place. These two procedures reflect the core logic of the HHA protocol as described in Section 4.1.3.

C.1 Event Handlers for FLD Protocol

Event Handler for PACKET_GENERATED Event

```
let the event be $e$ ;
let the event happened at node $i$ (extracted from $e$) ;
construct a new packet at node $i$ with next sequence number for $i$. Let us call this packet $P$ ;
for each outlink $i_s$ of node $i$
    if outlink $i_s$ of node $i$ is busy
            append $P$ to SEND_BUFFER[$i_s$] of node $i$ ;
    else
            let the outlink $i_s$ of node $i$ is connected with the inlink $j_r$ of node $j$ ;
            put packet $P$ on the receive buffer for inlink $j_r$ at node $j$ ;
            place a new PACKET_ARRIVED event in the event-list with the following:
                $type$=PACKET_ARRIVED, $time$=$sim\_clock$+$P.size$,
                $node$=$j$, $link$=$j_r$, $success$=current state of link ($i$-$j$)
            mark outlink $i_s$ of node $i$ busy ;
            append $P$ to the SEND_BUFFER[$i_s$] of node $i$ ;
place a new PACKET_GENERATED event in the event-list with the following fields:
    $type$=PACKET_GENERATED,
    $time$=$sim\_clock$+PACKET_GENERATION_INTERVAL,
    $node$=$i$
```

Event Handler for BLOBS_MOVE Event

```
let the event be $e$ ;
let the event happened for error blob $i$ (extracted from $e$) ;
move error blob $i$ to a new location as suggested by the mobility model ;
for each event $g$ in the current event list that has type PACKET_ARRIVED
    let this event indicates a packet arrival from node $m$ to node $n$ ;
    if the new location of error blob $i$ blocks link ($m$-$n$)
        set $g.success$=false ; // indicates that the packet transmission was unsuccessful
place a new BLOBS_MOVE event in the event-list with:
    $type$=BLOBS_MOVE, $time$=$sim\_clock$+ERROR_BLOBS_INTERARRIVAL,
    $node$=$i$ // indicates that the event occurs for error blob $i$
```

Event Handler for PACKET_ARRIVED Event

```
let the event be $e$ ;
let the event happened at inlink $i_r$ of node $i$ (extracted from $e$) ;
extract the packet from receive buffer $i_r$ of node $i$. Let us call this packet $P$ ;
let the inlink $i_r$ of node $i$ is connected with the outlink $j_s$ of node $j$ ;
delete a packet from the head of SEND_BUFFER[$j_s$] of node $j$ ;
if SEND_BUFFER[$j_s$] of node $j$ is empty
        mark outlink $j_s$ of node $j$ not busy ;
else
        put the next packet in SEND_BUFFER[$j_s$] of node $j$ on the receive buffer $i_r$ of node $i$ ;
        place a new PACKET_ARRIVED event in the event-list with the following fields:
                type=PACKET_ARRIVED, time=sim_clock+size of this new packet,
                node=$i$, link=$i_r$, success=current state of link ($j$-$i$)
update simulation variable total_bits_transmitted to reflect the transmission of the packet $P$ ;
if $e.success$ = false
        return; // return from handler if the packet transmission was corrupted
if $P$ is not a DATA packet // only DATA packets are allowed in FLD
        return;
let the source and the sequence number of the packet be $s$ and $seq$, respectively ;
if the touple ($s$, $seq$) is present in the FLOODING_TABLE of node $i$
        return ; // this packet has been seen before
insert the touple ($s$, $seq$) into the FLOODING_TABLE of node $i$ ;
if node $i$ is the sink
        update simulation variables packets_delivered and total_delay to reflect
        successful delivery of this packet and to include its delay in the output metric ;
else
        for each outlink $i_s$ of node $i$ except the outlink that connects to node $j$
                if outlink $i_s$ of node $i$ is busy
                        append $P$ to SEND_BUFFER[$i_s$] of node $i$ ;
                else
                        let outlink $i_s$ of node $i$ is connected with the inlink $k_r$ of node $k$ ;
                        put packet $P$ on the receive buffer for inlink $k_r$ at node $k$;
                        place a new PACKET_ARRIVED event in the event-list with:
                                type=PACKET_ARRIVED, time=sim_clock+P.size,
                                node=$k$, link=$k_r$, success=current state of link ($i$-$k$)
                        mark outlink $i_s$ of node $i$ busy ;
                        append $P$ to the SEND_BUFFER[$i_s$] of node $i$ ;
```

C.2 Event Handlers for HHA Protocol

Event Handler for PACKET_GENERATED Event

```
let the event be $e$ ;
let the event happened at node $i$ (extracted from $e$) ;
construct a new packet $P$ at node $i$ with next sequence number for $i$ and with the following:
        REMAINING_PATH=PRIMARY_PATH for node $i$
        LINKS_DOWN=nil
place $P$ on PACKET_BUFFER of node $i$ ;
if PACKET_BUFFER of node $i$ was empty before inserting $P$
        call FORWARD_ONE_PACKET(node $i$) ;
place a new PACKET_GENERATED event in the event-list with the following fields:
        type=PACKET_GENERATED,
        time=sim_clock+PACKET_GENERATION_INTERVAL,
        node=$i$
```

Event Handler for BLOBS_MOVE Event

```
let the event be $e$ ;
let the event happened for error blob $i$ (extracted from $e$) ;
move error blob $i$ to a new location as suggested by the mobility model ;
for each event $g$ in the current event list that has type PACKET_ARRIVED
    let this event indicates a packet arrival from node $m$ to node $n$ ;
    if the new location of error blob $i$ blocks link ($m$-$n$)
        set $g$.success=false ; // indicates that the packet tx was unsuccessful
place a new BLOBS_MOVE event in the event-list with:
    type=BLOBS_MOVE,
    time=sim_clock+ERROR_BLOBS_INTERARRIVAL,
    node=$i$ // indicates that the event occurs for error blob $i$
for each node $j$ in the grid
    for each link $j$-$k$ from node $j$
        if the link $j$-$k$ is not inside or intersects any of the current error blobs
            mark link $j$-$k$ up in the LINK_STATUS structure of node $j$ ;
```

Event Handler for ACK_TIMEOUT Event

```
let the event be $e$ ;
let the event happened at outlink $i_s$ of node $i$ (extracted from $e$) ;
mark this outlink down in the LINK_STATUS structure of node $i$ ;
if $forwarded\_on\_original\_link$=0 for node $i$
    if PACKET_BUFFER of node $i$ is not empty
        call FORWARD_ONE_PACKET(node $i$) ;
    else
        print error and exit ;
else
    let $x$ be the number of outlinks of node $i$ that are up according to its LINK_STATUS;
    if $x$=0
        place a new RETRY_TIMEOUT event in the event-list with:
            $type$=RETRY_TIMEOUT,
            $time$=$sim\_clock$+RETRY_INTERVAL,
            $node$=$i$
    else
        if PACKET_BUFFER of node $i$ is not empty
            call RECOMPUTE_PATH(node $i$) ;
        else
            print error and exit ;
```

Event Handler for RETRY_TIMEOUT Event

```
let the event be $e$ ;
let the event happened at node $i$ (extracted from $e$) ;
call FORWARD_ONE_PACKET(node $i$) ;
```

Event Handler for PACKET_ARRIVED Event

```
let the event be e ;
let the event happened at inlink $i_r$ of node i (extracted from e) ;
let us assume the following:
    inlink $i_r$ of node i is connected with the outlink $j_s$ of node j
    outlink $i_s$ of node i is connected with the inlink $j_r$ of node j
extract the packet from receive buffer $i_r$ of node i. Let us call this packet P ;
delete a packet from the head of SEND_BUFFER[$j_s$] of node j ;
if SEND_BUFFER[$j_s$] of node j is empty
    mark outlink $j_s$ of node j not busy ;
else
    let the packet at the head of SEND_BUFFER[$j_s$] of node j be Q ;
    if Q is a DATA packet
        set Q.tx_end_time=sim_clock+Q.size ;
    put the packet Q on the receive buffer $i_r$ of node i ;
    place a new PACKET_ARRIVED event in the event-list with the following :
        type=PACKET_ARRIVED, time=sim_clock+Q.size,
        node=i, link=$i_r$, success=current state of link (j-i )
update simulation variable total_bits_transmitted to reflect the transmission of P ;
if P is a DATA packet
    if e.success=true
        // send ACK to node j:
        build a new packet of type ACK called P_ACK with the following:
            source and sequence of P_ACK identical to that of P
            set P_ACK.tx_end_time = P.tx_end_time
        append P_ACK to the SEND_BUFFER[$i_s$] of node i ;
        if outlink $i_s$ of node i is not busy
            put the packet P_ACK on the receive buffer $j_r$ of node j ;
            place a new PACKET_ARRIVED event in the event-list with:
                type=PACKET_ARRIVED,
                time=sim_clock+P_ACK.size,
                node=j, link=$j_r$, success=current state of link (i-j)
            mark outlink $i_s$ of node i busy ;
        // sending ACK complete
        if node i is the sink
            if this packet P has not been seen before
                add this packet to the list of packets seen so far ;
                update simulation variables packets_delivered
                and total_delay ;
        else
            place P on PACKET_BUFFER of node i ;
            if PACKET_BUFFER of node i was empty before inserting P
                call FORWARD_ONE_PACKET(node i) ;
    else // e.success=false
        place a new ACK_TIMEOUT event in the event-list with the following:
            type=ACK_TIMEOUT, time=P.tx_end_time+ACK_INTERVAL,
            node=j, link=$j_s$
else if P is an ACK packet
    if e.success=true
        let R be the packet at the head of PACKET_BUFFER of node i ;
        if source and sequence number of R matches with that of P
            delete one packet from the head of PACKET_BUFFER of node i ;
            if PACKET_BUFFER of node i is not empty
```

```
                call FORWARD_ONE_PACKET(node $i$) ;
            else
                print error and exit ;
    else
            place a new ACK_TIMEOUT event in the event-list with the following:
                type=ACK_TIMEOUT, time=P.tx_end_time+ACK_INTERVAL,
                node=$i$, link=$i_s$
```

Procedure FORWARD_ONE_PACKET (argument: node *i*)

```
let the packet at the head of PACKET_BUFFER of node $i$ be packet $P$ ;
// find the first packet in the PACKET_BUFFER that is not too old:
while (sim_clock - generation time of packet $P \geq$ TTL)
    delete packet $P$ from the PACKET_BUFFER of node $i$ ;
    if PACKET_BUFFER of node $i$ is not empty
        let the packet at the head of PACKET_BUFFER of node $i$ be packet $P$ ;
    else
        return ;
let the next node in $P$.REMAINING_PATH be node $j$ ;
let the outlink $i_s$ of node $i$ is connected with the inlink $j_r$ of node $j$ ;
if the link ($i$-$j$) is up according to LINK_STATUS structure of node $i$
    make a copy of $P$, let's call it $Q$ ;
    remove node $j$ from $Q$.REMAINING_PATH ;
    append the links that are down according to LINK_STATUS of $i$ to $Q$.LINKS_DOWN;
    update $Q$.size to reflect the new size of the packet ;
    append $Q$ to the SEND_BUFFER[$i_s$] of node $i$ ;
    if outlink $i_s$ of node $i$ is not busy
        set $Q$.tx_end_time=sim_clock+$Q$.size ;
        put the packet $Q$ on the receive buffer $j_r$ of node $j$ ;
        place a new PACKET_ARRIVED event in the event-list with:
            type=PACKET_ARRIVED, time=sim_clock+$Q$.size,
            node=$j$, link=$j_r$, success=current state of link ($i$-$j$)
        mark outlink $i_s$ of node $i$ busy ;
    set forwarded_on_original_link=1 for node $i$ ;
else
    let $x$ be the number of outlinks of node $i$ that are up according to its LINK_STATUS ;
    if $x$=0
        place a new RETRY_TIMEOUT event in the event-list with:
            type=RETRY_TIMEOUT, time=sim_clock+RETRY_INTERVAL,
            node=$i$
    else
        call RECOMPUTE_PATH(node $i$) ;
```

Procedure RECOMPUTE_PATH (argument: node i)

```
let the packet at the head of PACKET_BUFFER of node $i$ be packet $P$ ;
create a copy of the current topology ;
remove from this topology those links that are indicate down by LINK_STATUS
of node $i$ and LINKS_DOWN field of packet $P$ ;
apply shortest path algorithm on the new topology to find a new shortest path from $i$ to sink ;
if no such path exists
    place a new RETRY_TIMEOUT event in the event-list with:
        type=RETRY_TIMEOUT, time=sim_clock+RETRY_INTERVAL,
        node=$i$
    return ;
let the newly computed path indicates that the next node from $i$ to sink is node $j$ ;
let the outlink $i_s$ of node $i$ is connected with the inlink $j_r$ of node $j$ ;
make a copy of packet $P$, let's call it packet $Q$ ;
set $Q$.REMAINING_PATH=newly computed path ;
append the links that are down according to LINK_STATUS of $i$ to $Q$.LINKS_DOWN ;
update $Q$.size to reflect the new size of the packet ;
append $Q$ to the SEND_BUFFER[$i_s$] of node $i$ ;
if outlink $i_s$ of node $i$ is not busy
    set $Q$.tx_end_time=sim_clock+$Q$.size ;
    put the packet $Q$ on the receive buffer $j_r$ of node $j$ ;
    place a new PACKET_ARRIVED event in the event-list with:
        type=PACKET_ARRIVED, time=sim_clock+$Q$.size,
        node=$j$, link=$j_r$, success=current state of link ($i$-$j$)
    mark outlink $i_s$ of node $i$ busy ;
set forwarded_on_original_link=0 for node $i$ ;
```